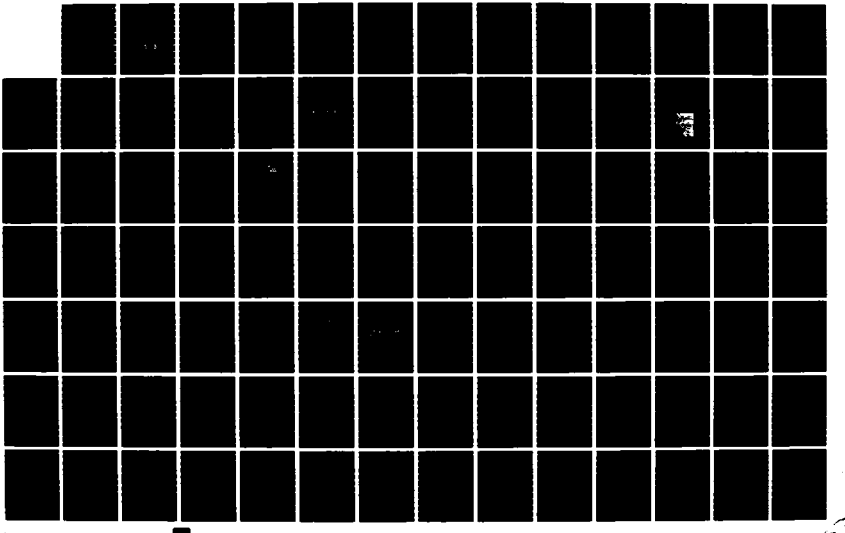
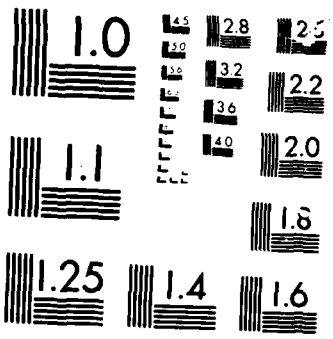
AD-A169 069 KNOWLEDGE-BASED PITCH DETECTION(U) MASSACHUSETTS INST 1/3
OF TECH CAMBRIDGE RESEARCH LAB OF ELECTRONICS W P DOVE
JUN 86 TR-518 N00014-81-K-0742

UNCLASSIFIED F/G 17/2 NL




1.0
1.1
1.25
1.4
1.6
1.8
2.0
2.2
2.8
3.2
3.6
4.0

Massachusetts Institute of Technology
Department of Electrical Engineering and Computer Science
Research Laboratory of Electronics
Room 36-615
Cambridge, MA 02139

Knowledge-Based Pitch Detection

Webster P. Dove

Technical Report No. 518

June 1986

This work has been supported in part by the Advanced Research Projects Agency monitored by ONR under Contract No. N00014-81-K-0742, in part by the National Science Foundation under Grant ECS-8407285, in part by Sanders Associates, Inc., and in part by an Amoco Foundation Fellowship.

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE A169069

1a. REPORT SECURITY CLASSIFICATION	1b. RESTRICTIVE MARKINGS
2a. SECURITY CLASSIFICATION AUTHORITY	3. DISTRIBUTION/AVAILABILITY OF REPORT Approved for public release; distribution unlimited
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE	
4. PERFORMING ORGANIZATION REPORT NUMBER(S)	5. MONITORING ORGANIZATION REPORT NUMBER(S)
6a. NAME OF PERFORMING ORGANIZATION (6b. OFFICE SYMBOL (If applicable)) Research Laboratory of Electronics Massachusetts Institute of Technology	7a. NAME OF MONITORING ORGANIZATION Office of Naval Research Mathematical and Information Scien. Div.
6c. ADDRESS (City, State and ZIP Code) 77 Massachusetts Avenue Cambridge, MA 02139	7b. ADDRESS (City, State and ZIP Code) 800 North Quincy Street Arlington, Virginia 22217
8a. NAME OF FUNDING/SPONSORING ORGANIZATION (8b. OFFICE SYMBOL (If applicable)) Advanced Research Projects Agency	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER N00014-81-K-0742
8c. ADDRESS (City, State and ZIP Code) 1400 Wilson Boulevard Arlington, Virginia 22217	10. SOURCE OF FUNDING NOS. PROGRAM ELEMENT NO. / PROJECT NO. / TASK NO. NR 049-506 / WORK UNIT NO.
11. TITLE (Include Security Classification) Knowledge-Based Pitch Detection	
12. PERSONAL AUTHOR(S) Webster P. Dove	
13a. TYPE OF REPORT Technical	13b. TIME COVERED FROM _____ TO _____
14. DATE OF REPORT (Yr., Mo., Day) June 198	15. PAGE COUNT 225
16. SUPPLEMENTARY NOTATION	
17. COSATI CODES FIELD / GROUP / SUB. GR.	18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)

19. ABSTRACT (Continue on reverse if necessary and identify by block number)

Many problems in signal processing involve a mixture of numerical and symbolic knowledge. Examples of problems of this sort include the recognition of speech and the analysis of images. This thesis focuses on the problem of employing a mixture of symbolic and numerical knowledge within a single system, through the development of a system directed at a modified pitch detection problem.

For this thesis, the conventional pitch detection problem was modified by providing a phonetic transcript and sex/age information as input to the system, in addition to the acoustic waveform. The Pitch Detector's Assistant (PDA) system that was developed is an interactive facility for evaluating ways of approaching this problem. The PDA system allows the user to interrupt processing at any point, change either input data, derived data, or problem knowledge and continue execution.

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT UNCLASSIFIED/UNLIMITED ☒ SAME AS RPT. ☐ DTIC USERS ☐	21. ABSTRACT SECURITY CLASSIFICATION Unclassified
22a. NAME OF RESPONSIBLE INDIVIDUAL Kyra M. Hall RLE Contract Reports	22b. TELEPHONE NUMBER (Include Area Code) (617) 253-2569 / 22c. OFFICE SYMBOL

DD FORM 1473, 83 APR EDITION OF 1 JAN 73 IS OBSOLETE.

SECURITY CLASSIFICATION OF THIS PAGE

This system uses a representation for signals that has infinite domains and facilitates the representation of concepts such as zero-phase and causality. Probabilistic representations for the uncertainty of symbolic and numerical assertions are derived. Efficient procedures for combining such assertions are also developed. The Normalized Local Autocorrelation waveform similarity measure is defined, and an efficient FFT based implementation is presented. The insensitivity of this measure to exponential growth and decay is discussed and its significance for speech analysis.

The concept of a history independent rule system is presented. The implementation of that concept in the PDA and its significance are described. A pilot experiment is performed that compares the performance of the PDA to the Gold-Rabiner pitch detector. This experiment indicates that the PDA produces 1/2 as many voicing errors as the Gold-Rabiner program across all tested signal-to-noise ratios. It is demonstrated that the phonetic transcript and sex/age information are significant contributors to this performance improvement.

Knowledge-Based Pitch Detection

by

Webster P. Dove

Submitted to the Department of Electrical Engineering
and Computer Science on May 9, 1986 in partial fulfillment
of the requirements for the Degree of Doctor of Philosophy
in Electrical Engineering

ABSTRACT

Many problems in signal processing involve a mixture of numerical and symbolic knowledge. Examples of problems of this sort include the recognition of speech and the analysis of images. This thesis focuses on the problem of employing a mixture of symbolic and numerical knowledge within a single system, through the development of a system directed at a modified pitch detection problem.

For this thesis, the conventional pitch detection problem was modified by providing a phonetic transcript and sex/age information as input to the system, in addition to the acoustic waveform. The Pitch Detector's Assistant (PDA) system that was developed is an interactive facility for evaluating ways of approaching this problem. The PDA system allows the user to interrupt processing at any point, change either input data, derived data, or problem knowledge and continue execution.

This system uses a representation for signals that has infinite domains and facilitates the representation of concepts such as zero-phase and causality. Probabilistic representations for the uncertainty of symbolic and numerical assertions are derived. Efficient procedures for combining such assertions are also developed. The Normalized Local Autocorrelation waveform similarity measure is defined, and an efficient FFT based implementation is presented. The insensitivity of this measure to exponential growth and decay is discussed and its significance for speech analysis.

The concept of a history independent rule system is presented. The implementation of that concept in the PDA and its significance are described. A pilot experiment is performed that compares the performance of the PDA to the Gold-Rabiner pitch detector. This experiment indicates that the PDA produces 1/2 as many voicing errors as the Gold-Rabiner program across all tested signal-to-noise ratios. It is demonstrated that the phonetic transcript and sex/age information are significant contributors to this performance improvement.

Thesis Supervisor: Alan V. Oppenheim
Title: Professor of Electrical Engineering

Thesis Supervisor: Randall Davis
Title: Associate Professor of the Sloan School of Management

Many thanks must surely be
for persons who were in the know.
For all their help and faith in me,
thanks Randy D. and AVO.

For others helpful all along,
through evenings late and sessions long.
My thanks to persons short and tall
to DSPG members all.

A special word is granted few
whose office shared and social glue
was help when I was feeling blue.
To Stephen, Doug and Cory too.

During all this tribulation,
who paid the remuneration?
A generous organization,
thanks to AMOCO Foundation.

My greatest gratitude in life
is for a blessing I have had
in trying times both good and bad.
For all her help I love my wife.

For my daughter

Table of Contents

List of Tables

List of Figures

CHAPTER 1

Introduction

Goals

In this thesis our goal was to develop techniques for combining symbolic and numerical knowledge in signal processing systems. We felt that systems for solving problems in which both kinds of knowledge could be found were dominated by either a symbolic or numerical approach, and that a more balanced application of symbolic and numerical knowledge would lead to better performance.

1.1. Knowledge-Intensive Problems

There are numerous signal processing problems in which the signals involved and the phenomena that underlie them do not all fit straightforward mathematical models. In most if not all of these problems, there is a lot of knowledge available, but this knowledge can't all be expressed in simple mathematical terms.

An example of a problem of this sort is tracking seagoing vessels. There are many pieces of applicable input information: ocean acoustic data, radar data, satellite data, sightings, course plans, ocean currents, atmospheric conditions, etc. There is knowledge about wave propagation, ocean thermal and salinity phenomena, mechanical vibration, structural acoustics, propeller ratios, biological noises, typical biological feeding areas, typical shipping lanes, behavior patterns of commercial and military vessels (both national and foreign), activity patterns on ships, theories of sound propagation, etc. Much of this information can be further qualified by knowledge about

specific vessels, captains, companies and world events.

Another problem of this sort is the recognition of requests given as speech to a computer. In this case the input information is primarily acoustic. The pertinent knowledge includes sampling and filtering issues, acoustic temporal and spectral properties of speech, phonetic and phonological properties of the language, properties of noise sources, the phonetic and phonological manifestations of various accents, dialects, and individual speakers, the grammar of the language, the semantics of the requests and the nature of the information available to satisfy them.

These are two examples of a wealth of problems of this type. Considering these problems it is apparent that some information can be thought of as numerical (e.g. noise power spectra), and some as symbolic (e.g. language grammar). It is our belief that systems must be capable of employing both types of knowledge effectively if they are to employ most of the knowledge.

1.2. Knowledge-Based Solutions

Depth versus Breadth

Systems to solve problems of this type can approach the problem of dealing with this knowledge in different ways. One approach is to take a small subset of the knowledge and build a system around it. An example might be a system that assumes a probabilistic model for the sound generated by ships, assumes a dynamic model for their motion, and computes an estimate of the trajectories of all vessels by choosing the scenario that maximizes the probability of receiving the signals that were in fact received. Such systems use a small subset of the available knowledge because the cost of computing such "optimal" estimates becomes prohibitive very rapidly as the com-

plexity of the model grows. This approach can be characterized as using a small amount of knowledge in a very powerful way, a "depth" approach.

Another approach for dealing with knowledge-intensive problems is to employ heuristics that make use of many pieces of knowledge. In this case, the system is unlikely to be provably optimal since the interaction of the components of the system would be too difficult to analyze. Such systems often don't have a specific mathematical model for the problem they solve. Their design is based on concepts of how the parts of the system should function and interact. Concepts derived from common sense, experience with other approaches to the task, or from a (more tractable) mathematical analysis of the subproblem. This approach achieves its performance through extensive application of the knowledge, it goes for breadth.

Symbolic versus Numerical

In the field of signal processing most systems for problems that do not require a symbolic result are dominated by the use of numerical knowledge. In part this is because many such systems emphasize depth, and it is only natural that the ideas that are used are chosen from the mathematically attractive aspects of the knowledge; such ideas can be manipulated with the powerful mathematical tools at the signal processor's disposal.

However, even in signal processing systems that emphasize breadth, the data is represented and manipulated in primarily numerical terms. In the majority of signal processing systems for speech pitch detection, one does not find any reference to phonetic, or linguistic aspects of pitch production (see chapter 2 for a description of such knowledge). The focus of systems written by the signal processing community is on representing the mathematical properties of signals and manipulating those

properties, not on modeling or manipulating representations of non-numerical phenomena that might underlie those signals.

Conversely, if one examines systems developed by the expert systems community for problems that have both numerical and symbolic components, one sees a tendency to use symbolic representations, and a preference for the use of symbolic processing as the primary means of problem solving. The HEARSAY[1] system for recognizing spoken database requests, and the SIAP[2] system for determining ship locations from acoustic data are both dominated by symbolic approaches. In both systems, numerical processing appears as an initial stage to convert the input information to a primarily symbolic representation, with little use of numerical representations or numerical processing thereafter.

Knowledge-Based Signal Processing

There does not appear to be any fundamental reason why a balanced approach to the use of knowledge is not possible with these problems, systems which would employ symbolic and numerical representations throughout, and made "equal" use of symbolic and numerical processing techniques to solve the problem. Such a balance, by virtue of its potentially greater repertoire of knowledge, could achieve better performance than that possible by employing one approach or the other.

The focus of knowledge-based signal processing (KBSP) is on learning how to build systems that employ a substantial amount of knowledge to solve knowledge-intensive problems; systems that freely make use of numerical and symbolic knowledge, numerical and symbolic processing methods. As was mentioned at the start of this section, our intent was to learn how to combine symbolic and numerical knowledge in signal processing systems. Specifically, we were looking for new

computer representations for numerical and symbolic information, ways to combine those representations, and other ideas that would contribute to building such systems.

The approach we took to achieve these goals was to select a problem that had a balance of numerical and symbolic knowledge, and build a system to solve it, placing particular attention on the development of new ideas in the system that would be applicable to other problems. The problem we selected was a variant of the pitch detection problem, and the system we built is called the Pitch Detector's Assistant (PDA).

1.3. A Modified Pitch Detection Problem

The conventional problem of pitch detection involves analyzing a digitized speech waveform, and producing an aligned voicing decision and f0 estimate. This thesis focused on a problem that was somewhat different because solving it was to be a catalyst for learning about combining symbolic and numerical knowledge, more than an end in itself.

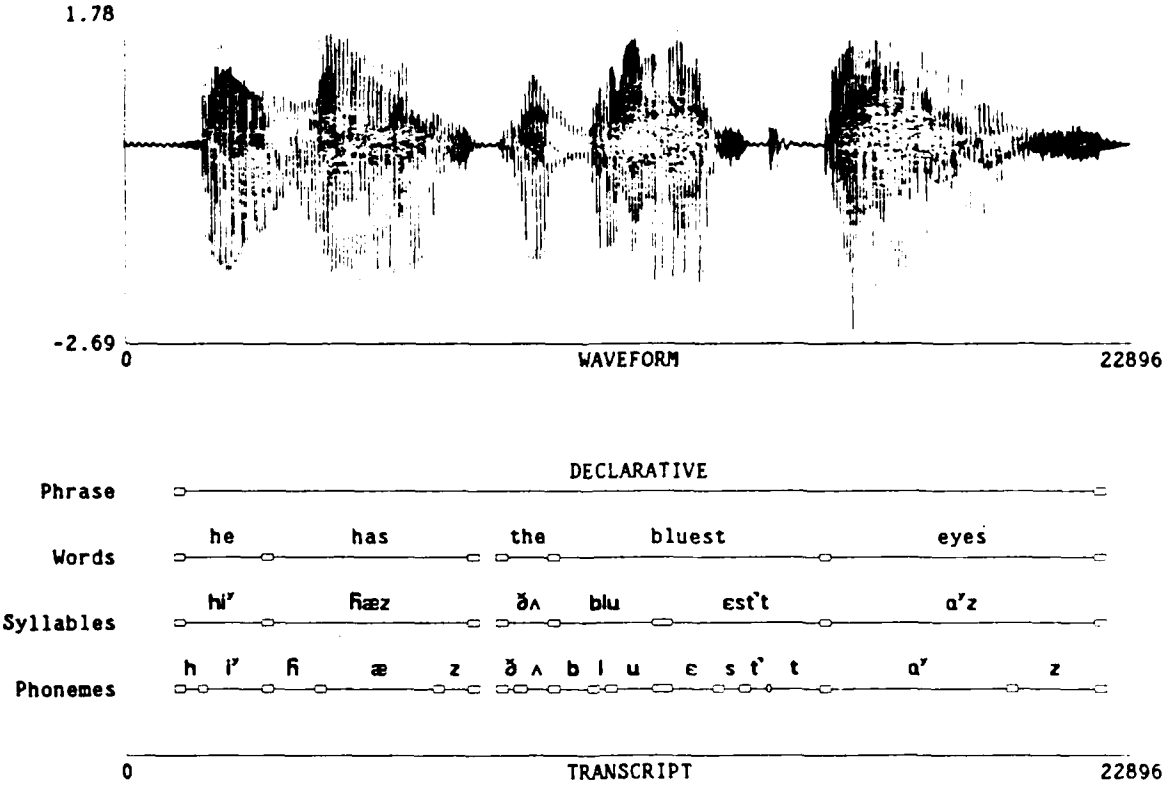
The pitch detection problem for this thesis was chosen to permit a balanced use of symbolic and numerical information, to encourage a balanced application of symbolic and numerical processing techniques, and to achieve this balance from the earliest stages of processing to the final output. This was accomplished in two ways: by changing the pitch detection problem statement, and by placing certain demands on the system that solved it.

The conventional pitch detection task is largely numerical. The input is a numerical sequence and one of the two outputs (f0) is numerical. In addition, conventional approaches to pitch detection describe time in numerical terms (by indexing with samples as opposed to using fundamental speech units such as phonemes or syllables).

Therefore, the problem of aligning voicing transitions and f0 estimates with the waveform is also in effect a numerical one.

Since the conventional problem is primarily numerical, our changes involved augmenting it symbolically. We made symbolic information available as input, information relevant to pitch and voicing determination. In particular, we supplied a phonetic transcript of the utterance (including word and syllable markings), as well as the sex and age of the speaker (either MALE, FEMALE or CHILD).

An example of these inputs is shown in figure 1.1. The waveform in the upper



Sex: Female

Figure 1.1 Inputs: Waveform, Transcript and Sex/Age

part of the figure is digitized at 10 kHz and represented in the computer as floating point numbers. The range of values are shown to the left of the vertical axis, and the range of indices are shown below the horizontal axis. The transcript is shown in the lower part of the figure. The four types of transcript marks (phrase, words, syllables and phonemes) are shown in the labeled vertical strata of the picture, and each mark is identified by a string of characters, with its extent depicted by the line below it. The widths of the boxes which border these lines signify the uncertainty with which those boundaries are known. Identical boxes which lie above one another are in reality just images of the same box as viewed from the different strata. The thin line which appears under most syllable marks depicts the "syllabic nucleus", the vowel or vowel-like phoneme around which that syllable is built.

Symbolic input of this form helps our goal of combining symbolic and numerical knowledge in three ways:

- This symbolic information balances the numerical information already present in the problem, so there are opportunities to combine the two.
- Providing this information as input can reduce the tendency for early processing to be dominated by a numerical approach, as has been the case with a number of expert systems for processing signals.
- Providing the transcript as input alleviated what would otherwise have been a major task: the analysis of the waveform to generate that information. Such a difficult symbolic analysis problem might have led to symbolic processing being dominant.

Though it is unusual to augment the pitch detection problem in this fashion, there are potential uses for a system that could solve it. These include the generation of pitch tracks for use in the enhanced reconstruction of archival speech material, pitch analysis for talking computer databases, and reference pitch tracks for testing other pitch detectors. Also, similar problem scenarios exist that are unrelated to speech.

One example is the enhancement of satellite imagery when road maps of the area are available.

1.4. A Pitch Detectors Assistant

Another influence on this thesis was our abstract picture of the computer system that was to solve it. We intended to investigate many different approaches to combining numerical and symbolic knowledge. However, there are limits on the complexity of a system that can be built in one try. Also, it was not clear at the outset exactly what approaches would be interesting, nor was it clear what approaches would be feasible in the framework of a thesis. The two alternatives in such a situation are to build many separate systems each of which demonstrates some new ideas, or to develop a single system over a long time period by adding new ideas incrementally. We took the latter approach.

While we did not intend expending all our effort trying to maximize the knowledge present in this system, it was clear that future systems of this type would be large and would probably be developed in an incremental fashion. Thus, anything that we learned about facilitating the extended development of such systems would be a useful contribution. Also, the amount of knowledge we incorporated into this system would depend on the ease of incorporation, and the more knowledge that was incorporated in our system, the more we would learn about the process and power of combining symbolic and numerical knowledge. For these reasons we chose to picture the system we developed as a Pitch Detector's Assistant (PDA).

This view places a premium on the ability to interact with and extend the capabilities of the system. The focus on extensibility is appropriate for any incrementally developed system, the focus on interaction helps the developer to analyze deficiencies

in its behavior that suggest the need for new knowledge, and helps the developer to debug any problems that occur when the system is being modified.

1.5. An Overview of the Contributions

This thesis makes several contributions related to programming symbolic and numerical knowledge. Some major results are enumerated below:

- We argue that converting numerical information to symbolic information on input, and using only symbolic processing thereafter is inadvisable. We suggest system architectures with which that can be avoided.
- We present a number of different ways that symbolic and numerical information can interact and give examples of such interactions drawn from the PDA system.
- We offer new representations for uncertainty in symbolic and numerical assertions, and new methods for processing uncertainty that are derived mathematically from a few basic principles and implemented efficiently in the PDA.
- We offer a new model for the representation of signals in systems that admits a broad class of signals (including those with infinite duration and those that are periodic) and makes the representation of temporal phenomena like linear-phase and zero-phase straightforward.

Besides results that pertain to the representation of numerical and symbolic knowledge in systems, there are results that pertain to signal processing (a new algorithm for measuring waveform similarity that is insensitive to waveform envelope), rule based systems (a rule system whose results are dependent on the currently active rules and data, but not dependent the order of entry of rules and data, nor rules and data that were active but have since been retracted), and pitch detection (the PDA is shown to outperform the Gold-Rabiner pitch detector[3] over a wide range of signal-to-noise ratios, and the symbolic information is shown to be a determining factor in the PDA's performance).

1.6. Organization of the Thesis

This thesis is organized into 7 chapters. The Introduction has presented the goals of the work and motives behind them, described and justified the specific problem we attempted to solve, and mentioned some of the important contributions of the work. Chapter 2 presents the knowledge applicable to pitch detection with brief descriptions of the ideas and references for further reading. Chapter 3 describes the general architecture of the PDA system, the specific knowledge that is represented in it and how that knowledge is made to work. Chapter 3 also describes the major components of the PDA from a conceptual standpoint. Chapter 4 discusses the major components of the system in detail and examines issues of implementation. Chapter 5 presents a comparison of the PDA with the G-R pitch detector. Chapter 6 discusses significant points about the implementation of this system and relates them to the general problem of representing numerical and symbolic knowledge in systems, and chapter 7 discusses some results that relate to signal processing, expert systems and pitch detection and suggests some directions for future work.

CHAPTER 2

Domain Knowledge

This chapter presents knowledge that bears on the pitch detection problem. This knowledge originates from two research areas: speech communications, and signal processing. From speech communications there is knowledge about the mechanics, acoustics and linguistics of pitch. From signal processing there is experience with solutions to this problem in the form of algorithms for pitch detection.

While the following sections describe what we learned about pitch, bear in mind that it was not all incorporated into the program that we built (chapter 3 discusses the knowledge that is actually incorporated into our program and how it is used). There are three purposes served by a chapter which presents the whole spectrum of pitch knowledge:

Provide a point of comparison.
: Since we feel that our program incorporates more knowledge than its predecessors, it is only fair to show all the knowledge so the reader can judge the significance of our accomplishment.

Illuminate the structure of the knowledge
: Much of the effort in building this system involved designing data structures and procedures to represent pitch and the phenomena which influence it. Exposure to the available knowledge should help the reader understand the rationale behind these decisions by clarifying the nature of the things to be represented.

Provide an archive.
: This chapter can serve related work by compiling references to the topics of pitch and its measurement into a single document.

2.1. Knowledge about Pitch Production

2.1.1. An Overview of the Oscillatory System

This is a brief presentation of the physiology of pitch production. A more detailed account can be found in a paper by Ohala[4] and the references cited therein. The human vocal system is depicted in an X-ray photograph and schematic diagram in figure 2.1[5]. From the lungs, a single passage passes through the vocal folds of the glottis and up to the velum. There it branches with one passage going past the velum to the nose and the other passage going past the tongue to the mouth.

The velum can be used to close off the passage through the nose (the nasal tract), so only a single passage extends from the glottis. The tongue, lips and jaw can be used

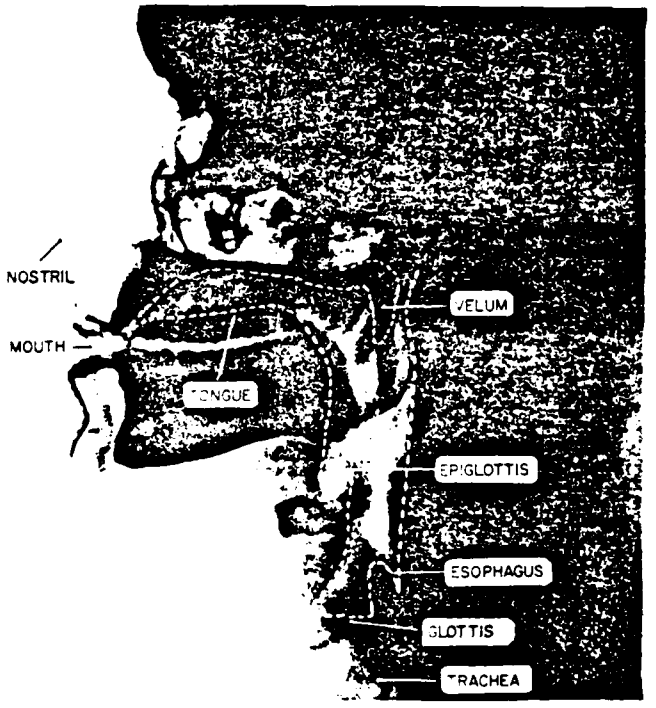


Figure 2.1 X-ray and schematic of the human vocal system

to vary the cross-section of the passage through the mouth (the vocal tract). By controlling the topology of the passages with the velum and the geometry of the vocal tract with the other articulators, humans implement a variable filter that controls the different sounds of speech. The arrangement of this apparatus that is used for a particular sound is called the "articulatory configuration" for that sound.

This system of passages is acoustically excited by three possible means: glottal pulses, turbulence generated by constrictions imposed on the air flow, and transients caused by abrupt pressure release.

Glottal Pulses

Glottal pulses are the (usually) periodic flaps of the vocal folds that give voiced speech its characteristic buzzy quality. These vibrations are driven by the air flowing between the vocal folds[6] like the vibrations of a trumpet player's lips. The rate of vibration is influenced by changes in the pressure drop across the vocal folds, changes in their tension, and adjustment of the average space between them. Some of these changes are caused by conscious attempts to manipulate f0 (the rate of glottal vibration). Others are the indirect effects of certain articulatory gestures as described below and in more detail in [4].

Turbulence

For certain articulatory configurations (e.g. "s") the nasal tract is blocked and the passage through the vocal tract is reduced to a very small opening somewhere along its length. This causes the particle velocity at the constriction to become very high, leading to turbulent noise generation. Other examples of sounds produced in this fashion are "f", "th" and "sh".

There are two terms for excitation of this kind: "aspiration" and "frication". When the constriction is made with the glottis, but the glottal configuration is such that vibration does not occur, then the resulting turbulent sound is called aspiration (as in the sound "h"). When the constriction is made with the tongue lips or teeth (as in the earlier examples), then the sound is called frication.

Transients

If the above constriction is carried to the point of complete closure, then pressure builds up in the oral cavity. Phonemes that involve such closure are called stops (e.g. "p" "t" "g"). When the articulatory configuration is changed after a stop and this pressure is released, two things can happen: turbulent noise and popping.

During release of a stop, the initial opening is small and turbulent noise (frication) usually occurs at the point of constriction (the so called "place of articulation"). As the opening grows there is less resistance to airflow through the constriction. The air velocity through the glottis increases, and that can in turn lead to either glottal vibration or aspiration depending on the state of the glottis.

The other possible occurrence during the release of a stop is the creation of a pop due to an abrupt change from zero to positive airflow. This event is not an inevitable consequence of stop release and is not considered a fundamental acoustic correlate of stop release. Nevertheless, it is important to realize that such a pop can occur because of the potential confusion between such pops and the low-frequency energy that usually indicates voicing.

2.1.2. General Properties of Vocal Excitation in Time and Frequency

From the above description we can see several modes of vocal excitation: glottal vibration, turbulence at the glottis (aspiration), turbulence elsewhere (frication), and possible pop transients during the release of stops. These modes of excitation have the following temporal and spectral properties.

Glottal Pulses

An approximation for the glottal waveshape during voiced speech is given in[7] as

$$g(n) = \frac{1}{2}[1-\cos(\pi n / N_1)] \quad 0 \leqslant n \leqslant N1 \tag{2.1}$$

$$= \cos(\pi(n-N_1)/2N_2) \quad N_1 \leqslant n \leqslant N_1+N_2$$

$$= 0 \quad \textit{otherwise.}$$

The time response and frequency spectrum of this approximation (using reasonable values of N_1 and N_2 is shown in figure 2.2.

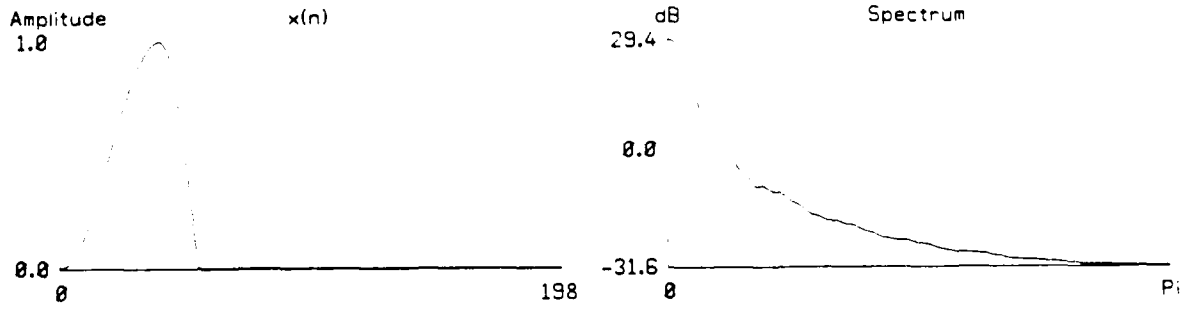


Figure 2.2 Glottal Pulse Approximation in Time and Frequency

Quasi-periodicity

During voiced speech, if the glottal source and vocal tract were perfectly stable and an infinite amount of data could be analyzed, then the frequency spectrum would consist of lines at multiples of the fundamental frequency of glottal excitation (f0). However, there is cycle to cycle variation in the glottal waveshape, temporal variation in f0, temporal variation in the vocal tract shape (and therefore its impulse response) and practical Fourier analysis must be done with a windowed data segment. Therefore, at best the spectrum of voiced speech consists not of lines at the multiples of f0, but of peaks at or near those multiples. A sample of a voiced speech waveform and its spectrum is shown in figure 2.3.

Noise Excitation

When there is turbulent excitation of the vocal tract, the frequency spectrum is one of a filtered white noise source. An example of this mode of excitation is shown in figure 2.4. The notable distinctions between these two sorts of excitation is that voiced excitation has substantial low-frequency energy and is generally repetitive in time

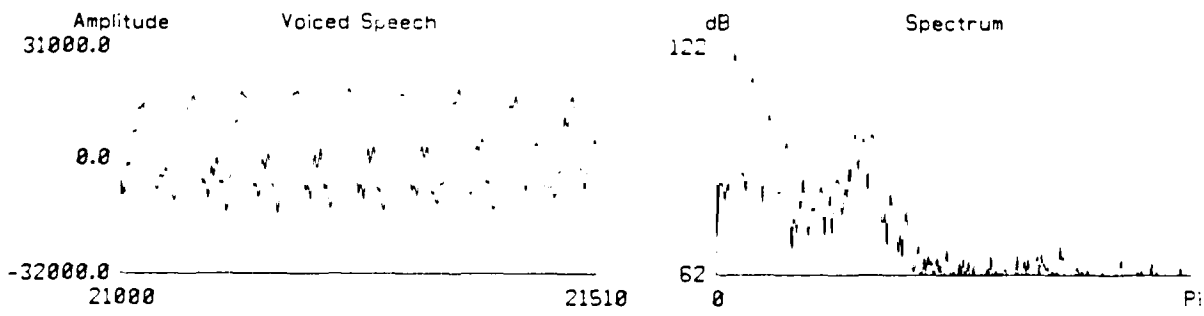


Figure 2.3 Voiced Waveform and Spectrum

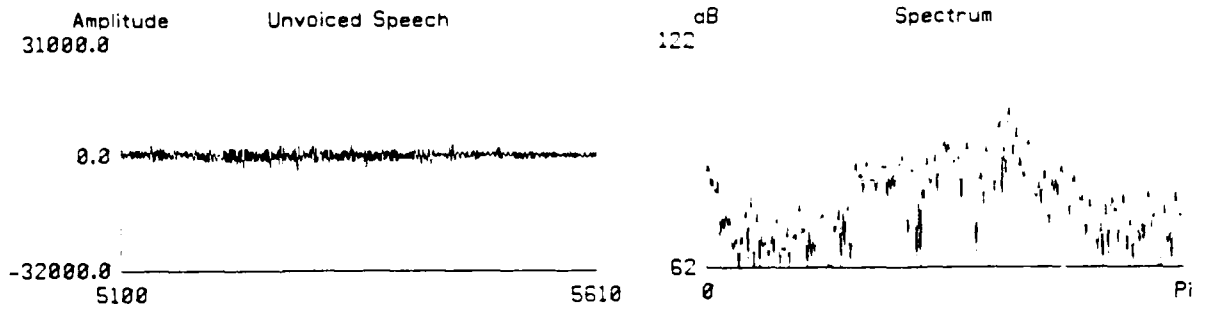


Figure 2.4 Unvoiced Waveform and Spectrum

(leading to a spectrum containing uniformly spaced peaks), whereas turbulent excitation has little low frequency power, no temporal regularity and therefore no uniform peak structure in the frequency domain.

Mixed Excitation

It is possible to have both glottal vibration and frication as in the sound "z". In this case the glottal portion of the excitation dominates in the low frequency range and the fricative portion in the high frequencies, leading to a spectrum that has a regular harmonic shape in some regions and an irregular noise in others[8].

Pops

The time and frequency response of a pop during stop release is shown in figure 2.5. The increased low-frequency energy can be seen as compared with the earlier spectrum from frication.

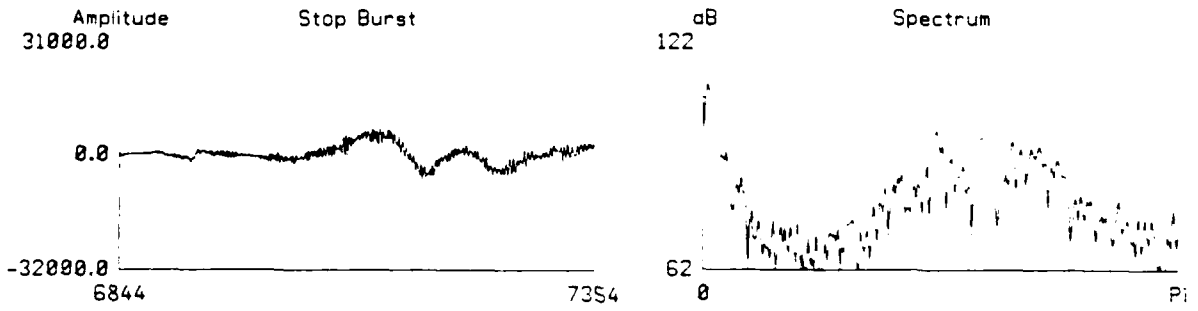


Figure 2.5 A Pop in Time and Frequency

2.1.3. Physiological Factors Influencing F0

Sex and Age

One of the most pronounced influences on f0 is the sex and age of the speaker. It is common knowledge and has been demonstrated in experiments[9] that female speakers and children have much higher nominal f0 values than males.

Variation in Excitation Spectrum with Voicing Intensity

For most voiced speech the vocal folds come completely together during part of the cycle. When voicing is intense and complete glottal closure is achieved, the abruptness of the closure leads to a fairly broadband excitation. However, when voicing is soft, complete closure may not occur during any part of the cycle, resulting in a smooth (narrowband) excitation waveshape. This effect is depicted in figure 2.6. The acoustic implications of this phenomenon is that phonemes which are "softly" spoken may have a depressed high frequency spectrum.

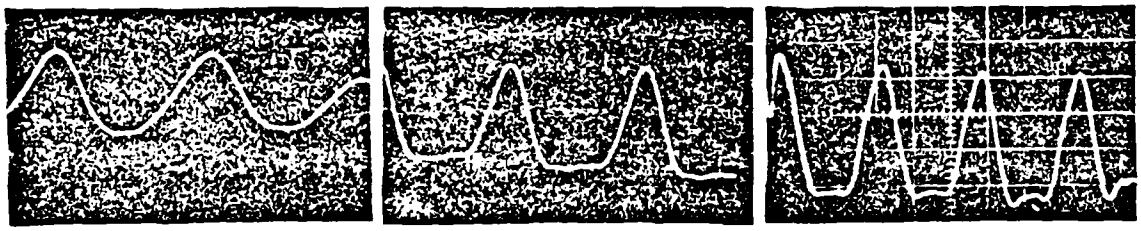


Figure 2.6 Variation in Glottal Waveshape with Intensity

Glottal Excitation During Closure

When voicing is sufficiently intense to completely close the vocal folds during part of each cycle, recent research[10] suggests that some vibration, and therefore some vocal tract excitation, may still take place. Acoustically, this means that even with a closed glottis, one cannot presume that the acoustic signal is solely determined by the impulse response of the vocal tract (as in some recent works on glottal inverse filtering[11, 12]).

Frication

Frication, achieved by forcing air through a constriction of the vocal tract above the glottis, leads to an increase in pressure between the glottis and the constriction. If this occurs during glottal vibration, that increased pressure may lower the rate of vibration of the vocal folds (due to the reduced pressure differential across them). Ultimately, this pressure will stop glottal vibration entirely.

If frication takes place without voicing and a rapid transition is made to a following voiced state (e.g. "fa"), then the rate of glottal vibration may be temporarily

elevated due to high airflow. This situation occurs because the glottis is wide open during the unvoiced fricative to prevent glottal vibration, and because of the pressure that has built up. When the fricative is released, while the glottis is closing in preparation for voicing, the high pressure and open vocal tract leads to a surge in the air velocity that elevates the initial glottal vibration rate.

Stops

Stop production also leads to a pressure rise above the glottis. Like fricatives, if glottal vibration is taking place, this can first cause a drop in f0 followed by the complete cessation of vibration. If a voiced stop is brief enough, vibration may continue during the entire closure phase of the stop.

Unvoiced stops can also temporarily elevate f0 in an immediately following vowel. These effects (and those for fricatives) are discussed and demonstrated in a paper by Lea[13].

F0 bias due to Articulatory Configuration

The position of the larynx is believed to play a role in f0 determination through the tension it places on the vocal folds. Since different articulatory configurations lead to different positions of the larynx, there is a bias in f0 that depends on the phoneme being uttered. This has been documented in a variety of experiments[14, 15]. At the present time, this information is only available for vowels.

Rate Limits

Since the f0 regulation system is mechanical, there are limits on the rate of change of f0 that can be achieved through active manipulation. These limits have been studied for both professional singers and laypersons[16, 17]. This information bears on

the types of interpolation that can be performed between pitch estimates. It also sets limits on acceptable rates of variation for plausible numerical pitch estimates.

An F0 model based on Physiology

One model for the behavior of f0 is based on the mechanical structures used to manipulate f0[17]. This model predicts that log(f0) can be modeled as the sum of responses of second order linear systems.

This model could potentially provide more constraints on pitch variation than the rate limits mentioned above, so it could potentially be more useful. However, applying it requires the determination of a driving function, and the designers of this model have yet to suggest how such an excitation might determined in the absence of a pitch track. The thrust of their research has been to perform the inverse problem of determining the model excitation from the pitch track.

2.1.4. Phonetic and Phonological Influences

F0 Effects

As was mentioned above, stops, fricatives and vowels can influence f0. Though such effects are physiological in origin, they occur only in specific phonetic contexts, so they can also be considered phonetic influences.

Stop Timing

Besides having an effect on f0, certain phonemes carry information about voicing onset time (VOT). Information about the timing of voicing with respect to phoneme position is not available for all phonemes. However, stops have been studied extensively[18].

Each stop can be subdivided into three phases:

- A closure phase during which the vocal tract is completely blocked and little if any sound is emanated (see "voiced bars" below).
- A burst phase during which fricative energy is produced by the rapid airflow through the opening constriction.
- A possible aspirative phase due to high airflow at the glottis.

These phases need not all be acoustically apparent. Voiced stops, for example, have little or no discernible aspiration. Also, when a stop is part of a stressed syllable (e.g. "repeal"), it is likely to manifest the acoustic behavior described above. However, when a stop is part of an unstressed syllable (e.g. "letter") these phases may not all be clearly in evidence.

In addition to the structural description of stops in terms of these phases, research also provides information on the timing of the phases. These times are strongly influenced by the surrounding phonetic context[18].

One use of such timing information is to subsegment the information in the transcript. Stops are transcribed in two parts: the closure and the release. By using stop timing information, the release can be further broken down into frication and aspiration. Identifying the fricated portion is useful because the pops that often accompany such frication can confound the detection of voicing. The surge in low-frequency power due to the pop may trigger detectors which rely on the presence of such power to indicate voicing.

Other Timing Effects

Other research on the influence of phonetic context on segment timing can be found in [19].

2.1.5. Syntactic Effects

A talker may use f0 and timing to distinguish certain words or groups of words in a sentence for syntactic purposes. This can involve elevated f0, pronounced variation in f0, and variation in the duration of segments[20].

Word Level

"Content" words may be distinguished from "filler" words in this fashion[21]. For example the noun subject of a sentence might be delivered starting with an unusual f0 rise that a preposition would not receive.

Phrase level

A noun phrase or verb phrase is often accompanied by a rapid rise in f0 followed by a gradual fall[22]. This seems to be a mechanism for helping the listener parse the sentence, and at least one method of automatically analyzing the syntax of spoken sentences has been based on this observation[13]. Also, a rising/falling f0 pattern on phrases is a feature of some speech synthesis programs (Dennis Klatt, private communication).

The overall sentence (if it is declarative and not a question) may also have this rising/falling f0 pattern imposed on it. The tendency for average f0 to fall during the course of a sentence is called "declination". While declination is a generally accepted aspect of f0 behavior in a declarative sentence, some believe that it is in fact a side effect of incremental downsteps in individual syllables, and not a global phenomenon[23].

2.1.6. Other Linguistic Factors

Stress and Prominence

When we speak a phrase, we employ two types of stress. The first is lexical stress. This is stress on some syllables of each word that is governed by the identity of the word alone (hence the term "lexical"). For example, "banana" is lexically stressed on its second syllable.

Lexically unstressed syllables are often underspoken. Some of the phenomena normally associated with a phoneme may be missing when it appears in an unstressed syllable[24]. This effect can turn a /t/ into a flap (a very brief stop-like sound) in the word "butter" and suppress the second vowel in "button" turning the pronunciation into "buttn".

The second type of stress is prominence. It is used to mark the significant words in the phrase being spoken and to emphasize the phrase structure. "Lets EAT here." is a phrase with the prominence on EAT. This phrase might be used to answer the inquiry "what should we do at the mall?". The phrase "Lets eat HERE" might be used to point out a specific store in which to eat within the mall. In both cases the purpose of the prominence is to point out the significant part of the communication. Prominence is usually attached to significant verbs or nouns in a sentence[23].

Prominence is intimately tied to f0 since one of the ways of achieving prominence is with significant transitions in f0. One of the important relations between it and lexical stress is that prominence is always placed on lexically stressed syllables.

Besides influencing the clarity of the acoustic manifestations of various phonemes, stress or prominence may systematically vary that manifestation. For example,

vowels which are stressed tend to be longer than those which are unstressed[20, 19].

Tune

The syntactic and linguistic influences on f0 have been incorporated into a theory for describing the approximate f0 trajectories used in English (ignoring consonental dip and articulatory bias)[23, 25]. This theory postulates a series of "tones" (much like the notes of a musical score) that are affixed to some of the lexically stressed syllables of a sentence. These tones specify a rise, fall or combined rise/fall at the location the associated syllable. A single "phrase tone" is associated with the last prominent syllable of the phrase and this tone specifies the behavior of f0 out to the end of the phrase. The set of tones for a sentence is called a "tune". The tune, together with speaker specific f0 parameters and information about the overall emphasis, is used to predict an average pitch contour for the sentence.

While this theory seems very attractive because it collects so many phenomena into a single description, there are some difficulties with using it. First, it is a developing theory and is changing in the face of new experimentation. Second, it depends on knowledge of the tones and their position in the sentence. While such information may be practical to acquire when experienced transcribers of tune are available, it is not easy at present.

2.1.7. Extra-Linguistic Factors

Speaking up

F0 is influenced when a talker attempts to improve the reliability of their communication by EMPHASIZING THEIR WORDS. To do this the speaker increases the loudness of his speech and increases the f0 variations that are used in it.

Emotion

Emotion also plays a part in determining the f0 of an utterance. Anger tends to be reflected by dramatic emphasis which leads to large f0 variations.

Jitter

Because the human glottis is a biological device, its oscillations exhibit noticeable cycle to cycle variation in period and waveshape. Experiments with the voicing of continuous tones has shown period variation of about .5% between successive periods, and total variation of 1.5%[26, 27]. There are no experiments on cycle to cycle period jitter in normal speech. However, a brief experiment (presented in chapter 5) suggests that 2% is a reasonable value to use for expected jitter. This figure is important in pitch detection because it defines when succesive period estimates can be considered the same.

Glottalization

Some speech waveforms indicate that the excitation is aperiodic glottal pulses with longer spacings than typical for periodic excitation. Spacings two or more times those expected for periodic vibration are not unusual. Aperiodically excited speech is called "glottalization".

Certain situations exhibit glottalization frequently. For example, if a word begins in an unstressed vowel, and the preceding word ends in a vowel then glottalization of the unstressed vowel is common (M. Bush private communication). Also, at the ends of sentences and places where f0 is unusually low, glottalization is likely[28]. (figure 2.7).

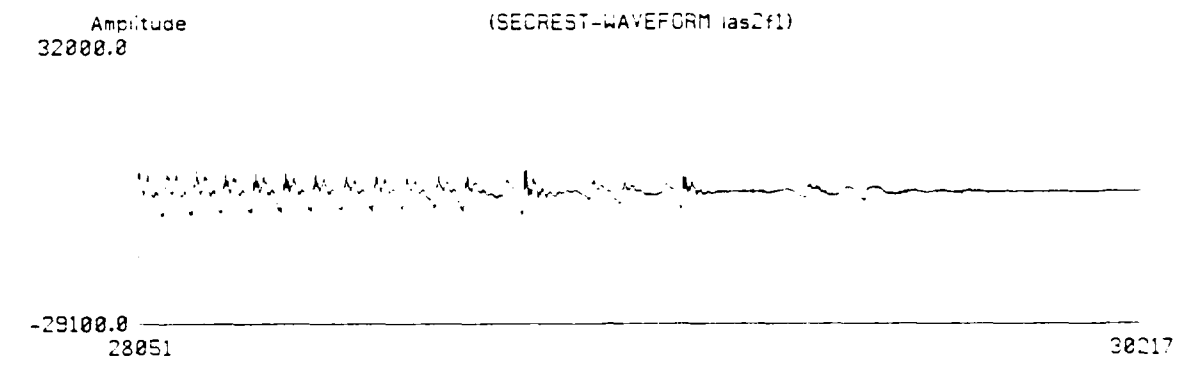


Figure 2.7 End of Sentence Glottalization

Diplophonia

Another unusual behavior of the glottis is its tendency to go into a mode of vibration that makes alternating pulses identical. This is called diplophonia. Although only occasional speakers manifest it constantly, it is not uncommon to find it in the speech of normal talkers at locations where glottalization would be expected (e.g. at the ends of sentences)[28].

Voice Bars

Voice bars is a term used to describe the parts of a speech spectrogram that appear when the sound emanating from the throat is recorded rather than the sound from the mouth or nose. This situation may occur during the closure portion of voiced stops.

While this does not truly represent an irregular voicing mode, the resulting spectrum can be almost sinusoidal. Thus if period estimates are to be maintained during this time interval, the method of detection cannot rely on the existence of several harmonics of f0.

2.2. Knowledge from Pitch Detection Algorithms

Since the 1940's pitch has been numerically determined for a variety of reasons. One goal is the storage and transmission of a compact representation of speech. Another is the recognition of speech so computers can be controlled by voice. Further goals include the diagnosis of vocal system pathology, the identification of speakers and the analysis of speaker stress. This work provides a body of knowledge about the behavior of various algorithms when used to detect pitch and voicing.

Signal processing algorithms for pitch detection all contain knowledge about the pitch detection problem. Some of that knowledge can be readily interpreted, as in the theory behind the basic algorithm. Other knowledge may not be as easy to interpret (e.g. limits on the bounds of program loops may signify an assumption about the range of permissible pitch values), or may be so bound up with the specific implementation that interpretation in the context of an abstract pitch detection problem is impossible (e.g. the choice of vote bias in the Gold-Rabiner decision scheme).

The remainder of this chapter discusses the knowledge represented in pitch detectors by presenting a general pitch detection idea and giving specific examples of pitch detectors which use it. We have tried in this discussion to avoid ideas that can't be related to the general problem of pitch detection, and only pertain to a particular method. Thus this is a catalog of pitch detection knowledge not a catalog of pitch detection algorithms. For a more detailed comparative discussion of various pitch detectors the reader should examine[29].

A definition of "periodicity"

The primary phenomenon involved in pitch detection is periodicity. By definition, a signal is periodic if a shifted version of the signal is identical to the

original signal for some shift (i.e. $\exists P : \forall n , x[n] = x[n+P]$). The "period" of a periodic signal is taken to be the smallest shift for which this is true.

In speech processing one describes the signal as "quasi-periodic" (meaning "approximately periodic"). In that context it is typical to think of "periodicity" not as an all or nothing property (as the dictionary would suggest), but as a variable quantity like period. There is no specific definition for the term "periodicity". Roughly speaking, we use it to mean: "How similar is the signal to a shifted version of itself?".

Periodicity in the Short Term

Speech is more than just almost periodic. One can assume that over intervals of a few periods, properties such as waveshape and period are relatively constant[30]. Thus, algorithms for determining period treat it as a slowly varying quantity that may reasonably be measured on sections of speech data that are a few periods in length.

A Common Program Structure

Programs for pitch detection generally operate in two phases. The first phase of the program transforms a section of the signal into a data structure from which potential period candidates and periodicity estimates can be readily derived. The second phase takes this data structure and either selects a single period choice, or declares the signal to be unvoiced. These two phases are applied on successive sections of the signal from left to right until all the data is exhausted.

These two phases can be further subdivided into four steps:

1a Preprocessing
: An initial analysis that changes the basic structure of the signal without producing a data structure from which the period and periodicity can be

easily measured. Typical examples of this are low-pass filtering to eliminate signal noise and possible aperiodic high frequency speech energy, and either clipping[31], frequency equalization[32], or linear prediction[33] to reduce the formant structure of the signal so the harmonic structure is more apparent.

1b Convert the preprocessed signal into the data structure from which the period and periodicity can be readily extracted.

2a Decision
Analyze the data structure and choose a period candidate.

2b Postprocessing
Derive a final period estimate on the basis of information about some or all of the period candidates from 2a. One example of this is non-linear smoothing[34], which can effectively eliminate individual gross errors in the pitch candidates from 2a.

A simple example of such a program would be one in which the first phase computes the short-time autocorrelation function of the signal section[6], and the second phase selects the largest peak of the autocorrelation function (other than at the origin) if it is above a threshold, and declares the speech unvoiced if not.

Because so many pitch detection programs have this form, it is a useful way to think of them for comparative purposes. However, this structure is not the only way of arranging such a program (the program developed for this thesis doesn't fit this model well). In the following sections we will (where appropriate) point out how a given algorithm can be decomposed in this way, and how this decomposition compares with those of other algorithms.

Period Range Limits

Many pitch detection algorithms scan over a finite range of periods (e.g. 3 ms to 15 ms). Searching for a period only in this range constitutes an assumption about the range of pitch periods in speech (the corresponding frequency range is 60 hz to 330 hz). While this is a reasonable range for most of the speech of normal male and female

talkers, it excludes the possibility of picking up unusually long glottalized periods or unusually high f0 values in speech from a child. For the speech corpus that was used for the experiments described in chapter 5 of this document, the above range would be too restrictive to permit accurate analysis. There were a few periods shorter than 3ms and periods longer than 15ms were not uncommon.

In terms of the pitch detection knowledge that this period restriction represents, it is fair to assume that most periods will fall within this range. However, it is inappropriate to assume that all periods will be so. Rather than unconditionally constrain the period estimate to lie within this range, a better approach would be to downgrade the confidence in a period estimate that lies outside. Such an approach would permit proper measurement of glottalized periods.

2.2.1. Temporal Similarity

"Temporal similarity" describes any pitch detection method that determines the periodicity and period of a signal by comparing it to a shifted version of itself. Since period changes over time in a speech signal, such a comparison must be made using windowed segments of data. In conceptually simplified terms, such an algorithm seeks the smallest P such that $\forall n \in N, x[n] = x[n+P]$. The set of samples N typically surrounds the location where the period is to be estimated.

The criterion for when $x[n] = x[n+P]$ and the determination of the set N are the two major things that differentiate the various methods which use temporal similarity as their basis.

An Average Magnitude Difference Pitch Detector

The average magnitude difference function (AMDF) pitch detector[35] uses temporal similarity to identify the periodicity and period of the signal. It creates a sequence that displays periodicity as a function of shift (candidate period) by computing the following expression for each integer value of P in the range [3ms 15ms]:

$$AMDF[P] \equiv \sum_{n \in N} |x[n]-x[n+P]| \quad . \tag{2.2}$$

where the set N contains 20ms of speech. This computation produces low values of $AMDF[P]$ when the signal is close to periodic with period P, and high values when it is not.

In this program, the procedure for selecting a particular shift value (as the final period estimate) involves a complex tracking algorithm. It uses information about the period and periodicity, as measured from the previous signal section, to guide the decision for the current section. This procedure is too complex and algorithm specific to be worth fully describing here.

The AMDF method fits the two phase description of pitch detection programs well. The first phase constructs the $AMDF[P]$ data structure without preprocessing. This sequence exhibits the lowest values at locations of strongest periodicity, and the location corresponds directly to the period estimate. The second phase is a (complicated) decision process, without any distinct postprocessing step.

An Autocorrelation Pitch Detector

The pitch detector designed by Dubnowski et. al. [31] also measures temporal similarity to determine periodicity and period. In this case, the comparison $x[n] = x[n+P]$ is performed using an inner product rather than an absolute

difference (as used by the preceding algorithm). The set N contains 30ms of speech, and the candidate period P runs from 2.5 ms to 20 ms.

This algorithm uses a preprocessor on the speech signal to spectrally flatten the signal for improved pitch detector performance. To perform this whitening, first the mean of the signal section is subtracted out (in case there is a DC offset), then the peak amplitudes over the first and last thirds of the section are averaged to estimate the clipping threshold C for that section. Finally, the incoming signal values are mapped to the values {+1,0,-1} using a threshold function, which is shown in figure 2.8.

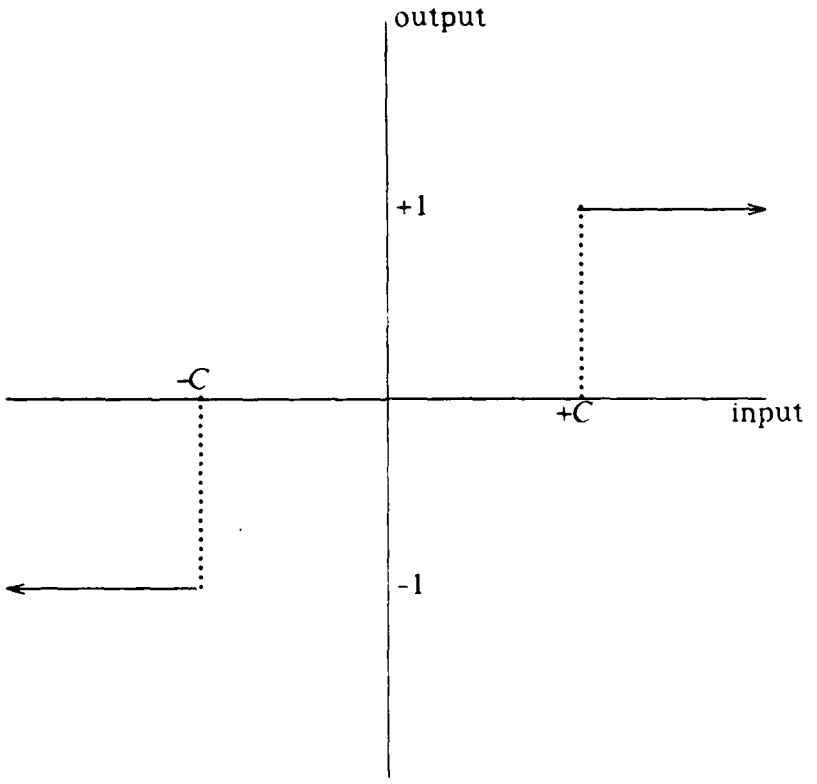


Figure 2.8 Clipping Function

To compute the inner product for non-zero shift P, the signal section $x[n]$ is treated as though it were padded with zeroes. This has the effect of applying a linear taper to the values of the inner product[6]. The weight applied to the zero shift inner product is 1.0, and the weight that would be applied to the 30 ms shift inner product (if it were computed) would be 0.

This sequence of inner products is the data structure resulting from the first phase of processing. Shift corresponds to period, and value corresponds to periodicity. Given this data structure, the shift (in the range from 2.5ms to 20ms) which yields the largest inner product is the period candidate. If that inner product fails to exceed 1/3 the height of the 0 shift inner product (which is always larger) or if the maximum amplitude for the section (before clipping) is below 1/20 of the maximum for the sequence, then the frame is declared unvoiced.

There are some new pieces of knowledge exposed here. One is that whitening has a potentially beneficial effect when one uses autocorrelation for computing periodicity with respect to shift. By flattening the spectrum of the speech, whitening reduces the tendency for the autocorrelation to have large values due to formant structure in the speech (resonances in the vocal tract) rather than due to periodic glottal excitation; large values that could lead to erroneous period candidates. The clipping system used by this pitch detection method is only one of a variety of means to achieve whitening, others include adaptive filter banks[32] and linear prediction[33].

Another new piece of knowledge involves the motive behind the linear taper on the inner product values:

> "The use of a linear taper on the autocorrelation function effectively enhances the peak at the pitch period with respect to peaks at multiples of the pitch period, thereby reducing the possibility of doubling or tripling the pitch-period estimate because of higher correlations at these lags than at the lag of the actual pitch period."

This exposes a fundamental problem in pitch detection. If a signal is periodic with period P, then it is also periodic with period $2P$. To guarantee (or at least encourage) the selection of the true period, the decision process must either emphasize the apparent periodicity at small shifts (as this algorithm does) or determine that no shorter plausible shift exists (as is done in the program written for this thesis).

The final new idea exposed by this pitch detector relates to the silence threshold. Algorithms that measure periodicity usually have some kind of normalization so they will work at all amplitude levels. In this algorithm, that normalization is embodied in the comparison with the zero shift inner product. Since the inner product at the shift which corresponds to one period and the inner product at zero shift are both proportional to the energy in the signal section, comparing them yields a procedure that is insensitive to scaling. When the speech becomes very soft so environmental and electronic noise sources dominate the signal, this type of algorithm can "lock on" to periodicities in the noise which have nothing to do with the speech signal.

To avoid such situations, this (and most other) pitch detectors employ a silence threshold and either declare the speech unvoiced at low amplitudes, or have a special "silent" voicing declaration for that purpose. Clearly the designers of these algorithms believe that voiced speech is only normally produced over a finite amplitude range (20 to 1 according to this particular method). While we have not found any speech research that corroborates this belief, it does seem to be a reasonable assumption.

2.2.2. Data Reduction

Digitized speech normally has many samples in each pitch period. Since pitch is considered to be slowly varying, this is many more samples than necessary to encode

the pitch information. Therefore it should be possible to convert the speech to a "reduced" representation that can still be used to estimate pitch, a representation that has fewer samples per second of speech. This is the meaning of "data reduction".

The attraction of data reduction is in eliminating the computational cost that goes with handling individual samples. Programs such as the autocorrelation method above must make tens of thousands of arithmetic computations for each 300 sample speech section. If each section could be reduced to a few numbers (e.g. one non-zero sample positioned at the start of each pitch period) then the computation savings would be considerable.

Gold-Rabiner

One pitch detector that relies on data reduction is the Gold-Rabiner (G-R) pitch detector[3]. This program first eliminates aperiodic high-frequency energy with a 900hz low pass filter. The resulting waveform is immediately reduced to a sequence of extrema (samples that are either larger than or smaller than their immediate neighbors) eliminating perhaps 90-95% of the samples. This extremal sequence is used to create six alternative streams of pulses by adding or subtracting adjacent extrema. Three of these streams are synchronized with the peaks in the original extremal sequence, and have pulse amplitudes as follows:

m_1 Each pulse is just the height of the corresponding extremal peak.

m_2 Each pulse is the height of the extremal peak plus the depth of the preceding valley.

m_3 Each pulse is the height of the extremal peak less the height of the preceding peak.

The other three streams are synchronized with the valleys in the original extremal sequence and are defined as follows:

m_4 Each pulse is just the depth of the corresponding extremal valley.

m_5 Each pulse is the depth of the extremal valley plus the height of the preceding peak.

m_6 Each pulse is the depth of the extremal valley less the depth of the preceding valley.

These definitions are illustrated in figure 2.9. and they yield six pulse streams with the same period as the original speech, but with many fewer samples per second.

These six pulse streams are fed to six identical trigger modules for a further reduction step. Each module produces a sequence of period estimates by running a blanking and decay circuit driven by its input pulse stream. Each time one of these circuits is triggered by a pulse, there follows a "dead time" during which triggering is prevented. After the dead time, a finite trigger threshold is set (initially to the height

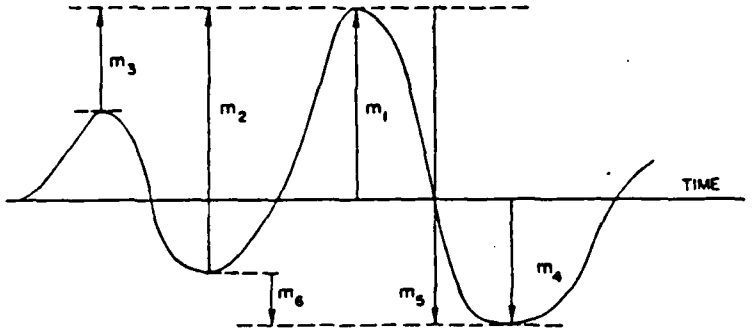


Figure 2.9 Gold-Rabiner Extrema

of the previous triggering pulse). This threshold decays exponentially with time until another pulse exceeds it, triggering the module and repeating the process. The dead time and the decay time are proportional to the average triggering time for that module. This second reduction procedure is depicted in figure 2.10, which depicts the action of a single module.

The output of each module is a sequence of period estimates. Each time a module is triggered, the time since the last triggering constitutes a period estimate. At any given time, the three most recent period estimates are available from each of the six modules. Every 10 ms a final period estimate is determined from this repertoire of 18 numbers.

The decision process which is the second phase of the G-R pitch detector is unique. It is presumed that each of the six modules is producing an equally reliable period estimate, and that only the most recent six estimates are viable choices for the final period estimate. First, a constituency of 36 "voters" is established from the 18 available period estimates. If each voter is labeled p_{ij} where i stands for the module and j for

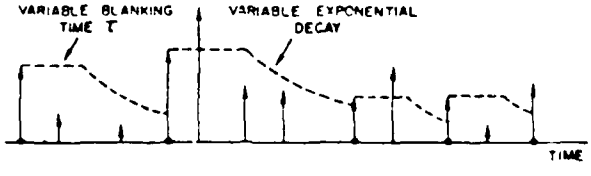


FIG. 4. Operation of the detection circuit which consists of a variable blanking time during which no pulses are accepted, followed by a variable exponential rundown

Figure 2.10 Trigger Modules

the voter from that module, then the voters are defined by figure 2.11. The three most recent period estimates, two pairwise sums and the triple sum, are all voters from that module.

The six candidates are the most recent estimates from each of the 6 modules. Four elections are held using four different tolerances for agreement between the candidate and the voter. For each election, a candidate gets one vote from each voter that agrees with the candidate within the specified tolerance. Since the tolerances for the four elections are different, and since the larger tolerances result in larger numbers of votes being cast, a bias is subtracted from each candidate's votes; a bias that grows with the tolerance and compensates for it.

If the candidate with the highest number of votes (including bias) has a positive median vote total across all elections, then that candidate wins, and that is the final period estimate. Otherwise (or if the peak amplitude of the section is less than 1/20 of the peak amplitude of the sentence) the section is declared unvoiced. Finally, a 3pt

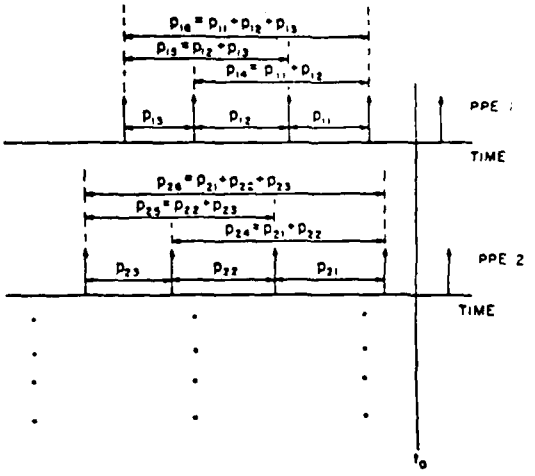


Figure 2.11 Period Estimates

median smoother is applied to the sequence of final pitch estimates to correct gross errors.

The G-R pitch detector is probably the most difficult of all pitch detectors to analyze for the knowledge it contains. Clearly there is knowledge behind the byzantine decision process, and the structure of the pulse streams and trigger modules, since it works so well. However, it is very difficult to tell what the ideas are, which apply to pitch detection and which to this specific approach, and which ideas are the most significant. When one speaks of knowledge that is "compiled in" to a program, this is a classic example.

This method demonstrates that period can be measured solely from extrema. This is a remarkable thing when you consider how much of a reduction that accomplishes, and how simple it is to do. For example, while zero crossings are a similar reduction and are also simple to find, there are no good examples of pitch detectors based on zero crossings.

This method assumes that the extrema of a speech signal fit a pattern that the trigger modules are designed to detect. Specifically, following a "trigger extremum" there is a period of uncertainty in extrema, followed by a period of decaying extrema followed by another trigger extremum.

In the use of sums of recent period estimates as voters we see the intent to correct for extraneous pulses triggering the modules. Such pulses divide what should be a single period estimate into two or more shorter ones. However, the original can be recovered by summing adjacent estimates as they have done. This particular knowledge is only significant for methods that are both attempting to use extrema and being plagued by extraneous pulses.

Perhaps the other most significant knowledge embodied in G-R is its use of redundancy. By having six trigger modules working on "different" views of the signal, and having a voting procedure to collect information from all those sources, the G-R pitch detector makes itself less susceptible to individual errors.

Data Reduction by Principle Cycle Analysis

This method[36] first band-pass filters the data to eliminate high frequency oscillation then makes a parametric representation of each half-cycles of the resulting waveform. This parametric representation is further reduced by analyzing the half-cycles and eliminating those that could not be the initial half-cycles of a pitch period. The goal is the isolation of the "principle cycles" which start each pitch period.

This system is unusual because it uses some notion of the structure of speech to determine its pitch measurement. Drops in amplitude are used to delimit syllables. An approximation of the average pitch during each syllable guides the final pitch estimation for that syllable. Thus this program embodies knowledge about the syllabic nature of speech and the fact that pitch variation within a syllable is small.

2.2.3. Harmonic Structure

When a signal is periodic, it has a spectrum composed of lines at the harmonics of the fundamental frequency. If it is quasi-periodic (as voiced speech usually is) then there are not lines but "peaks" instead. That is, one expects substantial spectral energy at or near the multiples of the fundamental, and little energy elsewhere. In the absence of a definitive statement of the nature of the quasi-periodicity, it is not possible to define the precise nature of these peaks.

Pitch Detection using Spectral Peak Positions

The fundamental frequency of such a quasi-periodic signal can be estimated by measuring the spacing of the peaks[37], adjusting a comb (or sieve) until it best fits the peak pattern[38, 39, 40], or looking for the greatest common denominator of the peak positions[41, 42].

Pitch detection by Harmonic Sum

These techniques[43, 44] use the entire spectrum to estimate the period rather than just the peak positions. In some sense, the distinction between these methods and those immediately preceding resembles the distinction between the data reduction time domain methods like G-R and the full time domain methods like autocorrelation.

Essentially, these methods take the inner product between the spectrum (or the log spectrum) and an impulse train. The spacing of the impulses that yields the largest projection becomes the fundamental frequency estimate. The largest estimate typically occurs when the impulses align with the peaks in the harmonic spectrum.

Cepstral Pitch Detection

Since the speech signal can be approximated as the convolution of a periodic pulse train with a low time-width filter function, the cepstrum can be used to turn the convolution of these two signals into a sum. This yields a low-time region containing the details of the vocal-tract response and a high-time region consisting of pulses located at the period of the speech and its multiples. By determining the spacing of those pulses or the initial pulse position one can estimate the period. This is the basis for the cepstral method of pitch detection[45]. We include it in the section on spectral methods because the computation of the cepstrum depends on computing the spec-

trum.

2.2.4. Other Methods

Maximum Likelihood Pitch Detection

The assumption that speech is a perfectly periodic signal with additive white gaussian noise, leads to a procedure for maximum likelihood pitch detection[46, 47]. These algorithms create a periodic signal by convolving the speech with an impulse train (aliasing). The spacing of the impulses is adjusted to maximize the similarity between this periodic signal and the original, and the resulting spacing is taken as the period estimate. It can be shown[44] that these algorithms which are computed in the time domain are virtually identical to the harmonic sum spectral method mentioned earlier.

Pitch Detection from Prediction Residual

These methods[33, 48, 49, 50] use "linear prediction" to produce a residual signal from the speech. By using a short predictor (10th to 16th order), correlations in the signal that manifest themselves over short lags are removed (namely the correlations due to the vocal tract response). Once the short term correlations have been eliminated, it is possible to detect pitch simply by looking for peaks in the resulting residual signal.

2.2.5. Voicing Determination

Voicing may be determined numerically in a variety of ways, many of which do not involve pitch estimation. Some of the standard methods are: low frequency power, broadband power, periodicity, zero-crossing rate and predictability.

Low Frequency Power

Low frequency power measures rely on the fact that glottal vibration leads to low frequency energy over a substantial time interval. Other forms of excitation (e.g. frication) lead to high-frequency energy or (like stops) to very short duration broadband power. Thus a low-pass filter up to 600hz followed by some smoothing to eliminate the effects of clicks and stops can yield a useful voicing estimator.

Broadband Power

Broadband power measurements can be used to determine silent intervals if the background noise is low. Such a direct determination of silence can eliminate the wasted processing that would otherwise take place. Also, estimators that factor out power (such as zero-crossings) can be very erratic at low power. So silence detection should be coupled with them to prevent this spurious behavior from triggering incorrect voicing errors.

Periodicity

Periodicity measures make use of the fact that glottal excitation is usually periodic. This is often used as a voicing method in pitch detectors because the period is being measured anyway. Unfortunately aperiodic speech is not uncommon, so pitch detectors that rely solely on periodicity to make their voicing determination will be prone to errors on some sentences.

Zero-Crossing Rate

The zero-crossing rate of a speech signal is sensitive to glottal vibration because the low frequency components that result from such excitation push the signal far from zero for (relatively) long periods of time. If the signal is dominated by noise, the

small but rapid deviations due to the high frequency components cause many more zero-crossings to occur.

As was mentioned above, this measure must be accompanied by a determination that power is present in significant quantities. When there is little power, the rate of zero-crossings is highly dependent on the nature of the background noise and therefore should not be used as a means to determine voicing.

Predictability

When there is glottal excitation, the acoustic waveform can be modeled as an impulse-like sequence exciting a slowly varying filter. In this case, linear prediction algorithms can reduce the power in the signal dramatically. When the excitation is random noise, prediction is much less effective in reducing power. Thus, the effectiveness of a linear predictor can be used as a means of determining voicing.

It is also the case that linear predictor coefficients have different responses to voiced and unvoiced speech. In particular, the first coefficient of a linear predictor has been used to help determine voicing[51].

Pattern Recognition Approaches

Numerical pattern recognition approaches to voicing determination rely on a combination of the above measures and training with pre-marked speech to numerically define the criteria for distinguishing voiced speech from unvoiced speech[51, 52].

2.3. Conclusion

There is considerable knowledge available that pertains in some way to the pitch detection problem. For reasons discussed in chapter 1, we did not seek to exhaustively

include all of it in the program that was developed for this thesis. The following chapters discuss the knowledge used in that program and how that knowledge was employed.

CHAPTER 3

System Architecture

This is the first of two chapters describing the Pitch Detector's Assistant (PDA) program. In Chapter 3 we begin with a general overview of the program and its components. The remainder of the chapter breaks the system down first in terms of the problem structure (knowledge about voicing and f0 determination) and then in terms of the program structure (the rule system, the dependency networks etc.).

Chapter 3 describes the system in terms of the intent of the design and the ideas that motivated it. The following chapter focuses on the actual implementation of the program, dealing with practical problems and the engineering decisions that were necessary to translate the ideas into practice.

3.1.1. An Example of System Operation

Before discussing the design of the system we present an example of its operation. The input to the system (except for the speaker sex/age) is shown in figure 3.1. The waveform is shown in the upper part of the figure. It is digitized at 10 khz and represented in the computer as floating point numbers[1]. The range of values are shown to the left of the vertical axis, and the range of indices are shown below the horizontal axis. The transcript is shown in the lower part of the figure. The four types of transcript marks (phrase, words, syllables and phonemes) are shown in the labelled vertical strata of the picture. Each mark is identified by a string of characters, and its

[1]While the original data was represented in 16 bit fixed point, all subsequent numerical processing was done in 32 bit floating point notation.

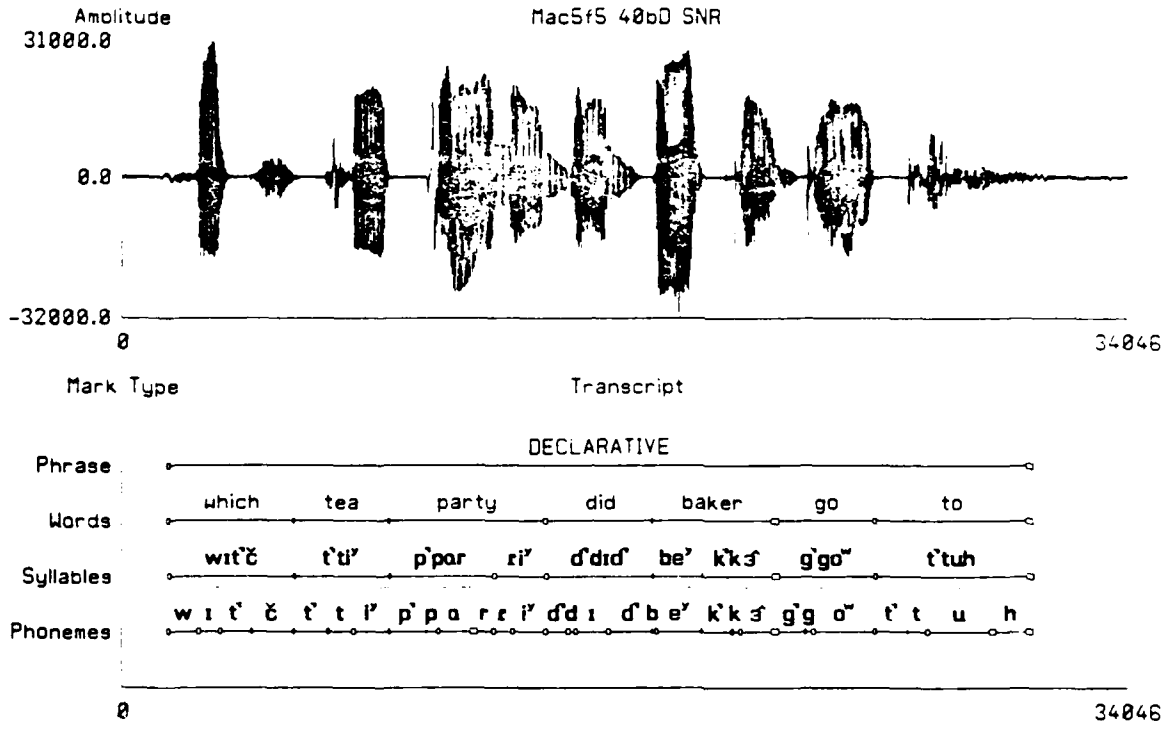


Figure 3.1 Inputs: Symbolic Transcript and Waveform

extent is depicted by the line below it. The boxes which border these lines signify the uncertainty with which those boundaries are known. In each case, the box depicts a Gaussian density whose mean and standard deviation are shown by the center and half-width of that box. Identical boxes which lie above one another are in reality just images of the same box as viewed from the different strata. The dotted line which appears under most syllable marks depicts the "syllabic nucleus" the vowel or vowel-like phoneme around which that syllable is built.

The results of processing are shown in figure 3.2. All four plots span time horizontally with the domain (in samples at 10khz) appearing below the horizontal axis of each plot. The first plot shows the "confidence" that the speech is glotally excited

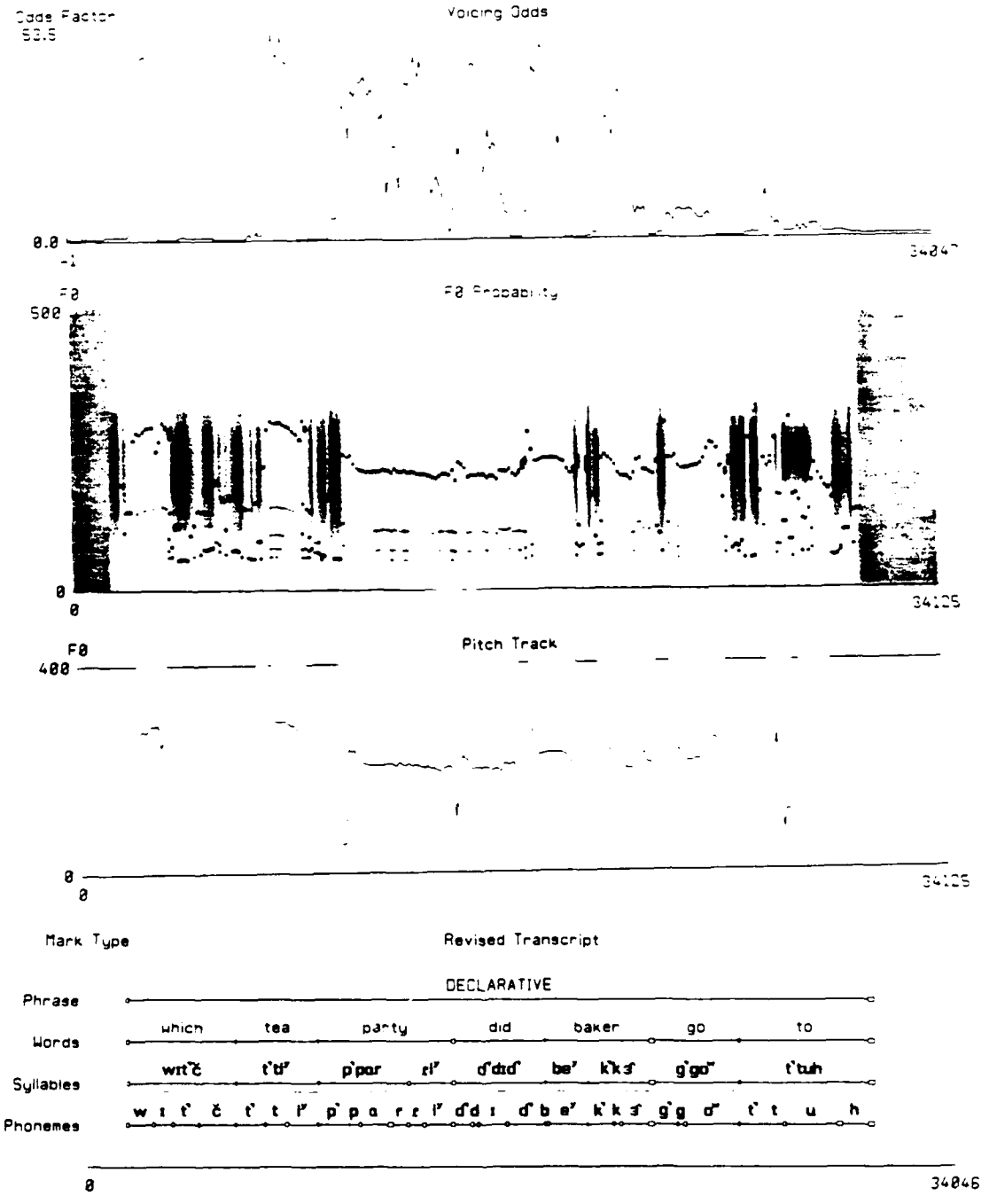


Figure 3.2 The Outputs of the PDA

(the voicing-odds-factor). The second is a pseudo-intensity plot[2] of the 3-dimensional surface which is the probability density of f0 as a function of time (the f0-probability-density). A vertical slice through this surface yields the probability density for f0 at that temporal location. The next plot shows the "revised" alignment of the phonetic transcript (where the depicted boundaries include information from numerical measurements) and the last plot is a conventional pitch track derived from the f0 probability density and the voicing confidence at each location.

3.1.2. The Symbolic Input

The design of the symbolic transcript input was based on the nature of the problem knowledge presented in Chapter 2. Phonemes are important because the phonetic context can be used to infer both pitch and changes in it. Syllables are the objects to which stress is attached. Words are important because of tendency for gaps to occur between them as opposed to within them, and because they are the means of indexing if one is to look up lexical stress. The phrase is the level at which one distinguishes question from declaration and the phrase delimits the outer bounds of the sentence.

While it may seem that providing a phonetic transcript would essentially solve the voicing decision problem, there are several reasons why this is not so. First, the boundaries of phonetic marks lack sufficient precision. Typically, they are only given with accuracy to the nearest few centiseconds. Since each analysis frame is one centisecond, errors of several frames are possible. Second, one cannot reliably determine voicing solely from phonetic identity. Some phonemes (like /z/ before a pause) are likely to change from voiced to unvoiced over their duration. Others like /r/ may be

[2] The density of dots in this plot corresponds to probability density for f0 with black being a high probability density and white being low.

voiced when used at the beginning of a stressed syllable, and unvoiced in an unstressed context. Lastly, the phonetic transcript cannot be expected to be completely accurate. Missing or incorrectly transcribed phonemes are possible. For all these reasons, the phonetic transcript does not solve the voicing decision problem.

The phoneme marks in the transcript were made by trained phoneticians who were given a plot of the waveform, its (300hz bandwidth) spectrogram and the words in the utterance. The phoneticians were not permitted to listen to the sentence. Subsequently, additional marks were added to delimit syllables, words and the phrase. In addition, syllabic stress was indicated for each syllable (chosen from the values: UNSTRESSED, STRESSED and PROMINENT) based on listening to the sentences[3].

In marking the phonemes, the phoneticians were asked to indicate their region of uncertainty at the boundaries. This was an unfamiliar task for them and in many cases a phonetician would mark a boundary with just a line (the conventional way of marking). In these cases, the judgment was made by the author based on the distinctness of the indicators for that boundary in the spectrogram. In any case, a lower limit of uncertainty of .01sec was used since that was the approximate resolution of pencil marks on the spectrogram.

3.1.3. The Outputs

The choice of outputs shown in 3.2 reflects two motives. The first motive was to show in detail the process by which the PDA program came to its conclusions. The second was to produce not only the "answer" to the question (the pitch track), but

[3] These marks were added by the author because they were necessary input to the PDA and were not a part of the transcriptions provided by the phoneticians. As the author is not a trained linguist, these added marks are somewhat suspect. However the uncertainty represented by such a marking was felt to be a proper challenge for the system.

also to express the uncertainty that is implicit in that answer (e.g. the voicing confidence is displayed rather than just a voicing decision).

The first motive is a reflection of the nature of an assistant and the importance of "interaction" between it and the operator. When processing a sentence, the outputs guide the operator in providing or revising information when they feel that the program is "confused" in some portion of the utterance. The outputs also help the program developer to understand and correct deficiencies in the system.

The second motive is that such supplementary information as confidence or statistics is important to the user of the program. It is not uncommon for a signal processing program to supply only the answer with no other information. This appears to be uniformly the case for other pitch detectors. However, there are two reasons for including information about uncertainty (if it can be estimated inside the program). First, if the user is a person, such information helps them decide how to interpret the answer in the context of their problem. Second, such information can be useful if the results are used in later processing.

When more than one system component contributes an answer to a single question, it is necessary to combine those answers. When the number of contributors is fixed and small, the combination function can be designed based on the known identity of the contributors. The voting matrix of the Gold-Rabiner pitch detector is a good example of this. However, if, as in PDA, where there may be many changing contributors, it is awkward to have to rewrite the combination mechanism for each new configuration. One cure for this is to have all contributors use a common means of communicating their answers, a means that contains enough information for the combining to be done without knowing the contributors from which the answers came.

PDA uses the idea of developing a language for answers, and isolating the identity of the contributor from the contribution. This idea was used in the system for refining estimates of boundary positions between phonemes, the system for determining the answer to symbolic valued questions (such as voicing), and the system for determining the value of f0. This ability to isolate contributor from contribution seems crucial to the ability to incrementally develop a large system without having a detailed understanding of the interaction of all of its parts.

Thus we see in the choice of outputs the interactive nature of an assistant, the desire to express to the user both the answer and the confidence in it, and the need in a composite system to express both the answers to questions and the supplementary information needed to combine those answers.

3.2. Pitch Detection in PDA

Since the pitch detection problem breaks down into the subproblems of finding voicing and f0, the PDA was designed to solve these two subproblems separately then combine the results (see figure 3.3). The approach of both subsystems is similar and can be described as follows: make assertions, verify them, combine results. Before describing the voicing and F0 subsystems we discuss the structures on which they are based: assertions and rules.

3.2.1. Assertions

Much of the information used and generated by the PDA during the analysis of a sentence is represented in the form of "assertions". Conceptually, an assertion is a statement about some aspect of the problem with supplementary information about confidence or statistics. Assertions may support and be supported by other assertions

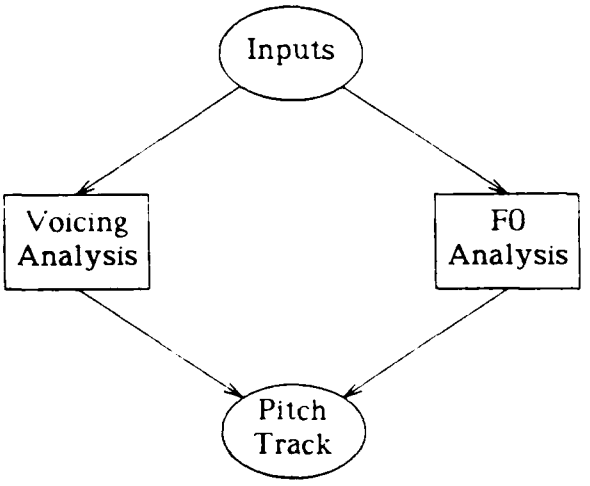


Figure 3.3 Basic Approach

and it is through this support that the confidence or statistics of a given assertion is determined.

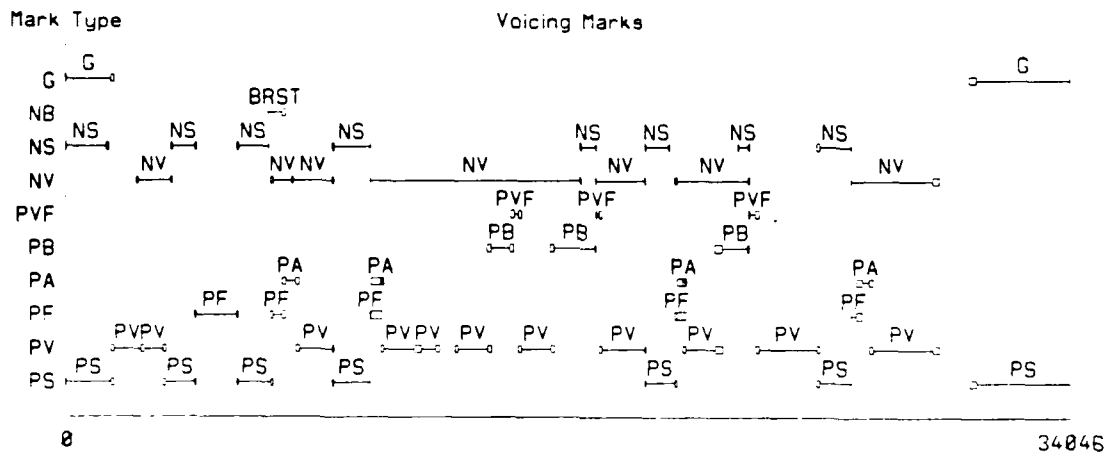
As assertions are added (or removed) from the system, the support they provide influences other assertions that are already present. The PDA is designed to propagate the influence of each change through the entire network of assertions. This means that the confidence in any given assertion is always kept up to date. A more detailed discussion of the implementation and operation of this network is presented in Chapter 4.

3.2.2. Voicing Related Assertions

In the voicing subsystem, most assertions take the form of symbolic statements about the utterance as a function of position. Each statement includes a description (e.g. "voiced" or "fricated"), a domain over which the assertion applies (some portion of the utterance), and a confidence (a numerical measure typically varying over the duration of the assertion. The phoneme, word, syllable and phrase marks in the input

(shown in figure 3.1) and the assertions generated by the PDA (shown in figure 3.4) are all assertions of this form.

The graphical depiction presented in these pictures shows the "domain" of the assertion as a line bounded by two boxes. The "description" is indicated by the string centered above the line[4]. For example, at the word level of the input transcript shown in 3.1 the descriptions are the words of the sentence.



BRST — Stop burst.
NS — Numerically measured silence.
NV — Numerically measured voiced.
G — Phonetically inferred gap.
PVF — Phonetically inferred voiced frication.
PB — Phonetically inferred voiced bars.
PA — Phonetically inferred aspiration.
PF — Phonetically inferred frication.
PV — Phonetically inferred voiced.
PS — Phonetically inferred silence.

Figure 3.4 Voicing Assertions

[4] In these figures, the confidence in the assertions is not shown. The confidence contours of some typical assertions are shown in a later section on combining assertions.

3.2.3. F0 Related Assertions

In the f0 subsystem each assertion takes the form of a sequence of probability densities for f0 over a portion of the utterance, and a sequence specifying the confidence that those densities are "valid". Any such assertion may be valid or invalid at any position within their domain. If the assertion is valid then the given probability density applies to f0 at that index. If it is invalid then there is no additional information about f0 there. The pseudo-intensity plot of f0 probability versus position in figure 3.2 displays the unique top level assertion (<FINAL-PITCH>) to which all other f0 assertions contribute. This assertion spans the entire utterance and specifies the probability density for f0 at each temporal position.

3.2.4. Combining Assertions

A system called the "knowledge manager" is responsible for updating the network of assertions whenever change occurs. This same system is also responsible for computing the confidence (and in the case of f0 assertions the f0 probability densities) of assertions. The details of the procedure for determining confidence are given in Chapter 4, but there are a few points mentioned here to help the reader understand the material that follows.

The confidence of each voicing assertion is expressed as a special kind of odds. The odds of some event can be derived from the probablility of that event by the expression $O(x) = P(x)/(1-P(x))$. Thus, the odds of an event ranges from 0 to ∞. Odds are usually expressed as a fraction. For example, 1/1 odds reads "1 to 1" and implies a 50% probability. Odds that are greater than one are called "in favor" and odds less than one are called "against". For example, one would say the odds are 9/1 in favor of voicing (90% probability) or 1/9 against voicing (10% probability).

Confidence in the PDA is expressed by what we call the "odds-factor". Specifically, the odds-factor of an assertion is the ratio of the "current odds" to the "a priori odds". Like odds, odds-factors also range from 0 to ∞. If an assertion is receiving no support (unlikely since the creator of the assertion typically supports it) it will have an odds-factor of 1.0, signifying that the assertion odds have not changed from their a priori value.

The reasons for using odds-factors to represent confidence are given in Chapter 4. The fact that they are used means that the assertions made by the PDA **must** be interpreted in the context of the a priori odds. If the odds-factor for a "voiced" assertion is 2.0 at some position, it does not mean that the odds are 2/1 in favor of voicing, it means that the odds are twice as high as they were a priori. If the a priori odds were 1/10 against, then they are now 2/10 against.

All voicing assertions (those derived from the transcript as well as those derived from the waveform) support a single assertion named <VOICED> which spans the entire utterance. The confidence of <VOICED> is determined by the net contribution from all supporters at each sample index, through a procedure that is described in detail in Chapter 4. The odds-factor for <VOICED> is the product of the "support-factors" from each contributor, with a support-factor of less than one lowering the confidence of <VOICED> and a support-factor greater than one raising it. It is the odds-factor for <VOICED> that is displayed in the output (figure 3.2) as the confidence of voicing.

To clarify this process, figure 3.5 shows the multiple contributions of support for <VOICED> over a subinterval of an utterance. The first plot shows the odds-factor for <VOICED> as a function of time, and the latter plots each show the support-

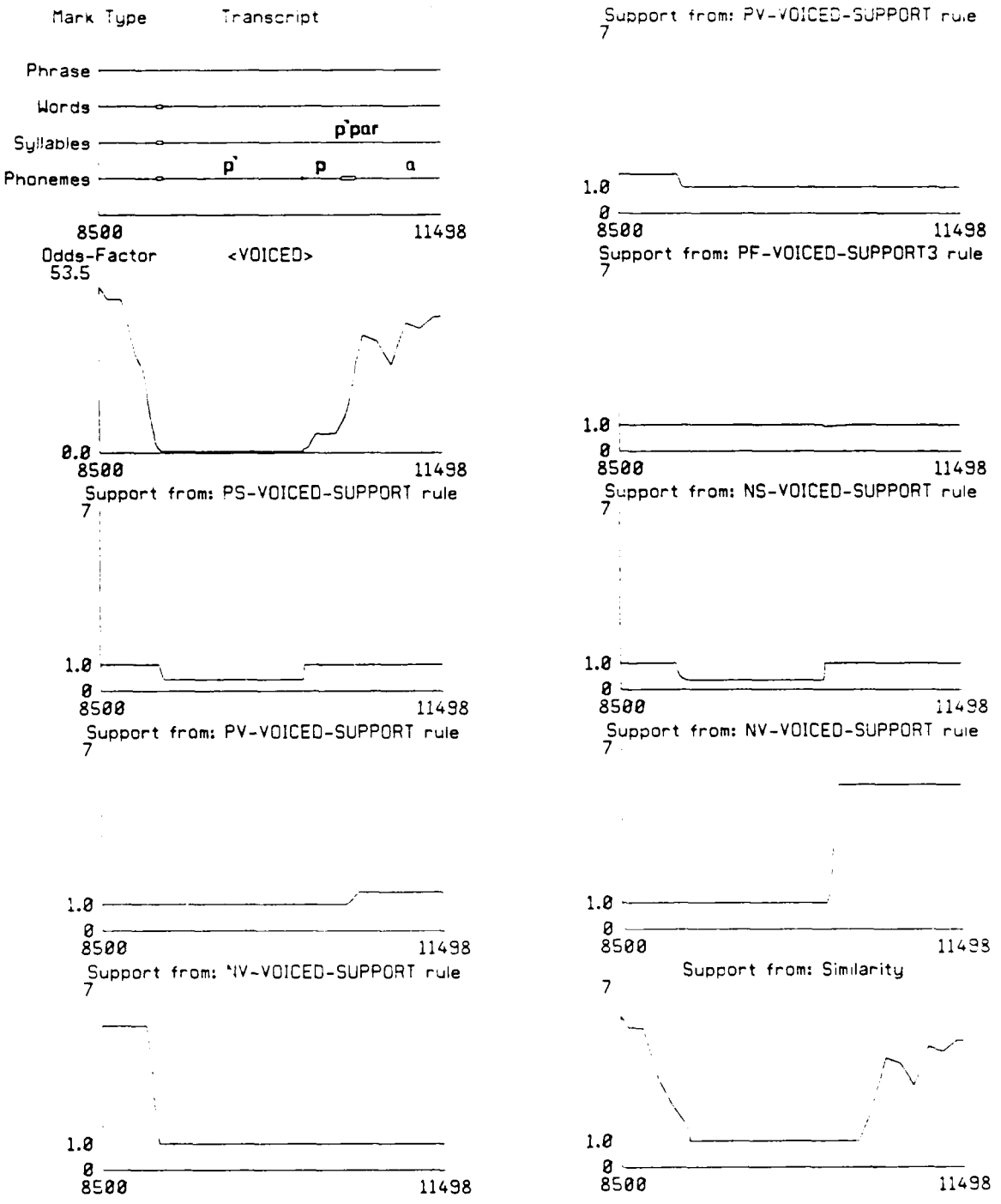


Figure 3.5 Support for <VOICED>

factor from a contributing assertion, labelled by the rule through which that assertion contributes support. For example, the support labelled "from rule <PV-VOICED-SUPPORT>" is from a phonetic voiced assertion (probably due to a vowel in the phonetic transcript). This support has a value larger than 1.0 where the vowel is located (since the vowel increases the confidence of voicing) and 1.0 elswhere (no contribution). On the other hand, the support labelled "from rule <NS-VOICED-SUPPORT>" has a value less than 1.0 where a numerical silence assertion is located (since silence decreases the confidence in voicing) and 1.0 elsewhere.

For f0, there is a similar <FINAL-PITCH> assertion. Each supporting pitch assertion contributes probability densities for f0 over its domain, and in <FINAL-PITCH> they are combined. In figure 3.6 there are windows showing the individual contributions of f0 probability density at a single position. The upper windows each represent the probability density contributed by a single assertion, this density reflects both the confidence and the basic density being asserted. The bottom window shows the final density that results from combining the upper ones. This is one slice of the f0 density shown in the pseudo-intensity plot in 3.2.

3.3. The Rules

The following explanation of rules should help the reader to interpret the rules given as examples in the following text. A more detailed description of the rule system is given later in this chapter, and details about the implementation are provided in Chapter 4.

This discussion and the explanation of the system that follows use essentially literal copies of the rules as found in the program. While that form of presentation may be difficult to understand at first, the rules are presented in this fashion to avoid

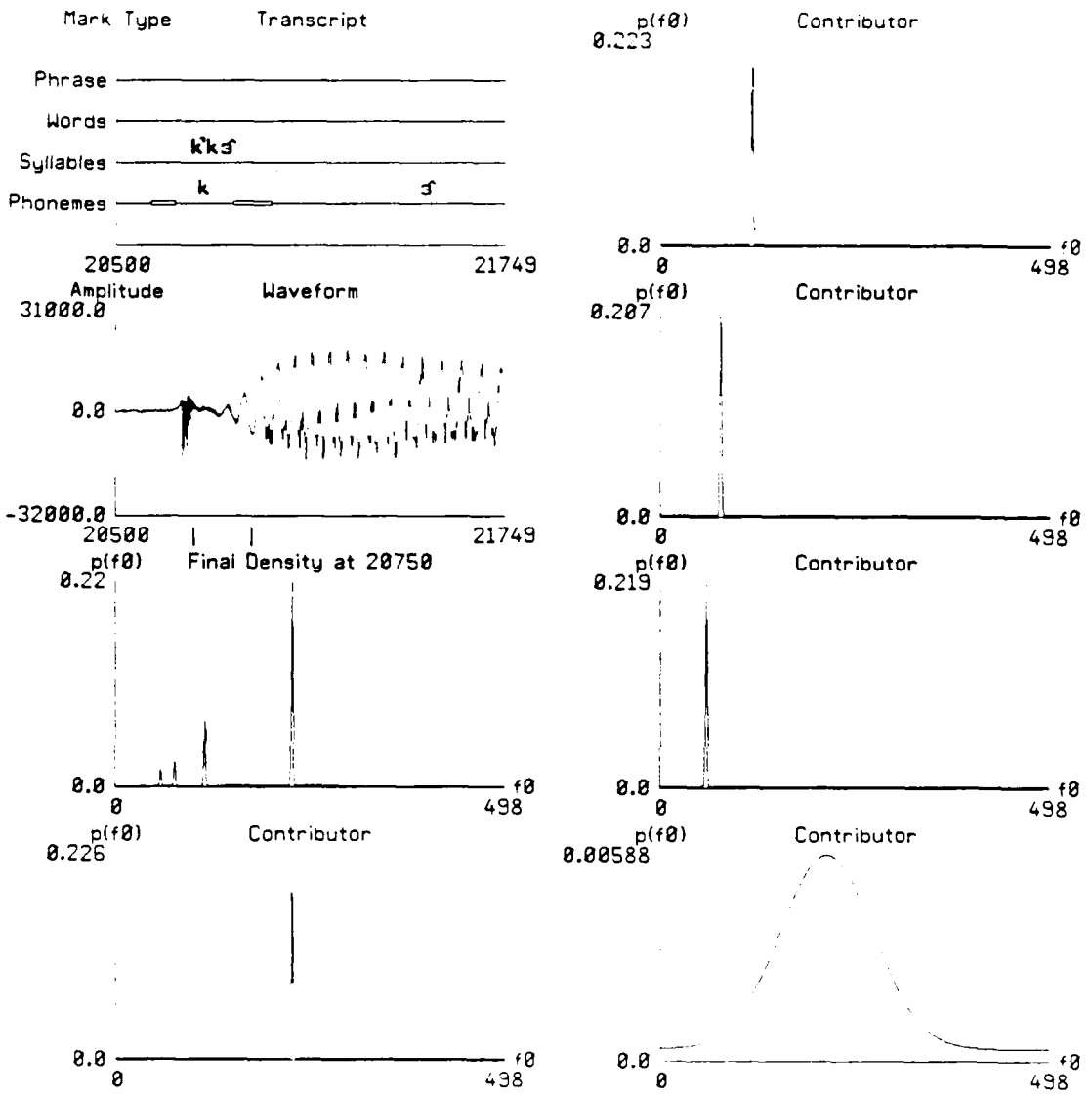


Figure 3.6 Support for <FINAL-PITCH>

the confusion that sometimes occurs when an "English translation" is presented in place of actual rules from the program. We hope the combination of rule text and explanation will serve both the casual reader and one intent on understanding the details of the program.

Consider the following rule for voicing determination:

```
(defrule <phonetic-voiced>
  CONDITIONS
  (type 'p-mark x)
  (same (voicing x) :voiced)
  (let start (start x))
  (let end (end x))
  ACTIONS
  (assert (phonetic-voiced start end) .8 .2)
  PREMISE ODDS
  (lambda (x) (odds x)))
```

The rule's name is "<phonetic-voiced>". The three uppercase symbols in the text of the rule precede the (only) three significant parts of any rule. The CONDITIONS part must be satisfied for the rule to "trigger". Triggering of the rule creates an object called a "binding" which both stands for that triggering of the rule and describes the unique association of assertions (and other objects) with rule variables that caused the triggering. Like an assertion, this binding has a confidence (represented as an odds factor). Unlike an assertion, this confidence is not computed by multiplying the support-factors of supporters. Instead, the supporters (which are a subset of the variable values) determine the odds-factor of the binding through the PREMISE-ODDS expression in the rule. Only variables which appear in the premise-odds expression are in fact supporters of the binding and contribute to its odds-factor. Finally, the ACTIONS part of the rule is a collection of lisp expressions that specify the changes to be made when the rule is "fired". These forms typically make other assertions, but it is also possible for the actions to cause the binding to give support to existing assertions. This is the way the rules mentioned in figure 3.5 provide support for the assertion <VOICED>.

For the preceding rule, the first form in the conditions declares the variable x, and specifies that it should be bound to new assertions of type p-mark. Each time a p-mark (phoneme-mark) assertion is made, the conditions of this rule will be checked

with the value of *x* bound to that assertion. The second form in the conditions specifies that the value of the expression (*voicing x*) be *:voiced*, thus the phoneme-mark bound to x must be for a voiced phoneme in order for the rule to trigger.

The next two forms are let forms. They both declare variables (*start* and *end* respectively) and assign them to the value of an expression (in this case, *start* = (*start x*) and *end* = (*end x*)). In part, these let clauses serve as a shorthand to avoid the need to rewrite expressions either in the conditions, the premise-odds or the actions. In addition, declaring any variable in the conditions (either with a type form or with a let form) signifies to the knowledge manager that this binding "depends" on the assertion or object that is the value of the variable.

Such dependency is discussed in detail in the section describing the knowledge manager, but basically it causes any *conditions* forms involving the variable to be rechecked whenever that variable value (object or assertion) is changed. Since the knowledge manager only monitors changes in variable values and not changes in expressions, assigning a new variable to an expression is the only way to make the rule responsive to that expression. For example, the expression (*start x*) returns an object that stands for the boundary at the start of the phoneme-mark.

Suppose the status of the boundary object was significant to the rule. Suppose it was important that the boundary object was also the start of a word. If at some point the rule was triggered and a binding was created, then it would be necessary for the knowledge manager to check the rule conditions if that boundary object were to change. Such a change might mean that the boundary object was no longer the start of a word, and that the binding was consequently invalid. The point is, the boundary object could change without the phoneme-mark changing. The only way to be sensi-

tive to changes in the boundary-object that is the value of the expression (*startx*) is to use a let clause to assign a new variable to it.

As it happens, in this rule the values of the variables *start* and *end* are not used in the conditions, so the let clauses only serve as shorthand. The premise-odds form is a lambda expression. In lisp this is a procedure definition, where the first list after the word λ is the argument list (x in this case), and the remainder of the form are expressions to evaluate, the last of which gives the value of the subroutine. In this rule, the value of the premise-odds form is just (*odds x*) the odds-factor of the phoneme-mark that triggered the rule. This becomes the odds-factor of the binding.

The reason this is a lambda expression is that it is necessary to have an argument list. Recall that the supporters of the binding are the values of variables that appear in the premise-odds form. Rather than have the PDA analyze the premise odds form to determine what variables are present, a list of the variables used is required. This both simplifies the task of the PDA and provides a check on the user's programming, since they must properly declare all variables and use them all in order not to generate warnings. Since the argument list and the body of the premise odds form were both already present, it was only logical to turn it into a common procedure declaration.

Eventually, when the rule is fired, the actions part makes a phonetic-voiced assertion that covers the same interval as phoneme-mark that triggered it. The two numbers in the (*assert* ...) form determine how the binding supports the phonetic-voiced assertion. A detailed description of the meaning of these numbers is given in Chapter 4, but basically the first number is the "probability of detection" and the second is the "probability of false alarm". The first number specifies the probability that the rule will be triggered if the actions are appropriate, the second specifies the

probability that the rule will be triggered if the actions are NOT appropriate. These two numbers are chosen by the system designer to represent their knowledge of the rules' behavior.

The following paragraphs recap the features and purposes of the three parts of a rule.

- CONDITIONS

  These are a set of lisp forms that must be satisfied for the rule to trigger. They specify the assertions that are necessary and as a side effect they bind rule variables to these assertions. These variables are passed to both the PREMISE-ODDS and the ACTIONS part of the rule.

  Many forms are just tests that must be satisfied (e.g. (*same* (*voicing x*) :*voiced*) is such a test). *Let* forms both declare variables and bind them, and *type* forms declare variables, specify that they should be bound to new assertions, and specify the type of assertion to bind them to.

  For the rule to be triggered, all tests in the rule must be satisfied. For each (*type* ...) form, an assertion of the proper type must be available and all test forms in the conditions must be true. On each iteration, the rule system takes each rule that is triggered and creates a binding that specifies the rule and the variable values that triggered it. Only one binding is created for each way assertions in the database can be associated with *type* variables in the rule. Thus each rule will fire exactly once on each satisfactory configuration of assertions in the database.

- ACTIONS

  These are a set of lisp forms that change the state of the system by altering existing assertions, or by adding new assertions to the database. New assertions made during the execution of the ACTIONS are "supported" by the binding in accordance with the probability of detection and probability of false alarm specified in the assert clause. The absence of such probability specifications implies the assertion will be support through some other means.

- PREMISE-ODDS

  The premise-odds λ expression determines the confidence in the binding that is created each time a rule is triggered. The variable values that support the binding must be declared in the argument list of the λ expression. The last form in the λ expression specifies the binding's odds-factor.

The rules which follow are not an exhaustive list of the rules in the PDA. Samples are taken from each of the types of rules. For example, though there are 10 rules for subsegmenting stops (corresponding to different phonetic contexts), only one such rule (<silence2> is presented here).

3.4. Determining Voicing

3.4.1. Overview

In PDA voicing is determined in two stages. The system first makes estimates of the voicing mode in various subintervals within the utterance. Then, by combining those estimates, it predicts the likelihood of voicing at each sample. The modes of voicing that PDA seeks to identify are: voiced, frication, aspiration, voiced-frication, voiced-bars and silence. All these voicing modes can be derived symbolically based on knowledge of the phonemes and their locations, and a subset of these modes (voiced and silence) can be found numerically by analyzing power levels in different spectral bands, and by looking for similarity in time.

Some phonetically derived assertions are first verified by checking for acoustic properties numerically. The presence (or absence) of these acoustic properties in turn affects the confidence of the voicing mode assertion. While no corresponding effort was made to verify numerically derived voicing mode estimates using phonetic information , there is no fundamental reason why that could not be done. However, given the nature of this project there was only time to implement one type of verification, and the phonetically derived assertions seemed to have the greater need for it.

The flow of analysis from the input to the voicing assertions and finally to <VOICED> is depicted in figure 3.7. Phoneme marks in the transcript are converted

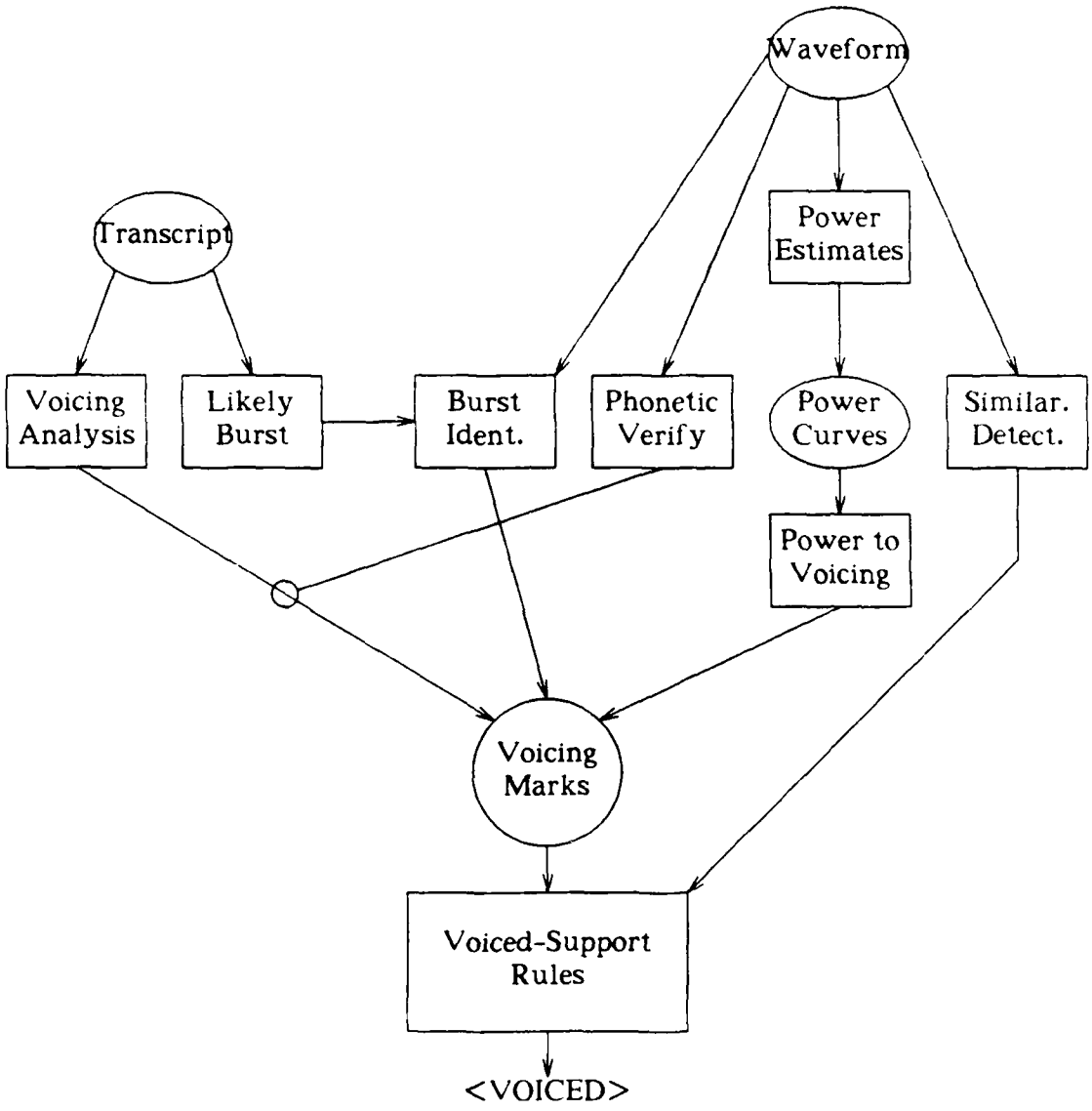


Figure 3.7 Determining Voicing

to voicing marks (Voicing Analysis). There is also a rule that spots the locations where bursts can be expected (Likely Burst) whose actions execute a burst identification procedure (Burst Ident.). This procedure analyzes the waveform and may produce a burst-mark if one can be located. From the waveform, broadband,

low-frequency (0-900hz) and high-frequency (2500hz +) power estimates produce a set of power curves that are numerically analyzed to generate voicing marks (Power to Voicing). The "similarity" of the waveform[5] is also measured (Similar. Detect.), and for many of the phonetically derived voicing marks, verification of their expected properties is carried out using the waveform (Phonetic Verify). Finally, the voicing marks that have been created and the similarity measurement of the waveform provide support to the unique assertion <VOICED> (Voiced-Support Rules) which is the voicing conclusion of the PDA.

3.4.2. Knowledge in the PDA

This section provides a brief description of each of the ideas used in PDA to determine voicing. The next section shows how those ideas were implemented.

Voicing from Phoneme Voicing

Most phonemes have inherent voicing (vowels are voiced, fricatives are frication, ...). In PDA there is a table that defines the properties (including voicing mode) of the various phonemes.

Voicing from Stop Aspiration

Stops have predictable burst duration and voice onset time (VOT). During the burst the voicing can be assumed to be frication, and during the rest of the VOT the voicing can be assumed to be aspiration.

[5] By this we mean how similar the waveshape is to the waveshape anywhere nearby. Strong similarity implies voicing.

Silent Gaps

If there is a gap from the start of the utterance to the first phoneme-mark or from the last phoneme-mark to the end of the utterance, the speech is likely to be silent there. If there is a gap between the end of one word and the start of another, the speech is likely to be silent there also.

Location of Stop Bursts

In the vicinity of an unvoiced stop release, the boundaries of the burst can be located by scanning outward from the peak in fricative energy.

Voicing from Sonorant Power

Large amounts of "sonorant" power (60-600hz) suggests that the speech is voiced.

Voicing from Broadband Power

Low Levels of broadband power signify silence.

Voicing From Similarity

If at any index it is possible to locate a nearby index where the waveform is "similar", then the speech is voiced.

3.4.3. Using the Knowledge

Voicing from Phoneme Voicing

The simplest example of voicing determination is the direct hypothesis of voicing from the identity of a phoneme. One rule which accomplishes this is shown in figure 3.8. This rule says "given a p-mark (phoneme mark) whose voicing is :voiced, make a

```
(defrule <phonetic voiced>
  CONDITIONS
  (type 'p-mark x)                ; $x$ is a phoneme-mark
  (same (voicing x) :voiced)      ; the phoneme must be voiced
  (let start (start x))           ; shorthand variables for $start$ and $end$
  (let end (end x))
  PREMISE ODDS
  (lambda (x) (odds x))             ; the binding confidence is that of
                                    ; the phoneme-mark $x$
  ACTIONS
  (assert
     (phonetic-voiced start end)  ; assert a voicing mark
     .8      ; probability of rule firing if it should (.8)
     .2))    ; probability of rule firing if it shouldn't (.2)
```

Figure 3.8 Voicing from Phoneme Identity

phonetic-voiced assertion over the interval of that p-mark"[6]. There is a similar rule for the following categories: frication, voiced-frication, voiced-bars.

The assertion support numbers of .8 and .2 represent moderate confidence that the mark given by the phonetician really warrants such an assertion (values of 1.0 and 0.0 would represent absolute confidence, and values of .5 and .5 would represent no confidence whatever).

Voicing from Stop Aspiration

An example of the determination of voicing from stop aspiration is shown in figure 3.9. This is one of ten rules that make frication and aspiration assertions from marks involving unvoiced stops. These are the most complex rules in the system[7]. This rule seeks phoneme clusters of the form

[6] A "mark" is our term for some annotation of the utterance.

[7] Their number and complexity reflect an interest in experimenting with the expression of knowledge within the PDA system more than the significance that this information may hold for voicing determination.

```
(defrule <silence2>
  CONDITIONS
  (type 'p-mark stop)  ; $stop$ is a phoneme mark
  (isa 'unvoiced-stop release stop)  ; it must be an unvoiced stop release
  (let left (left-neighbor stop)) ; the preceding phoneme-mark
  (isa 'stop closure left) ; must be a stop closure.
  (let 2nd left (left neighbor (left neighbor stop))) ; 2nd preceding
  (let right (right-neighbor stop)) ; following phoneme-mark
  (isa 's 2nd-left)   ; 2nd previous phoneme must be an /s/
  (nota 'semi vowel right) ; following phoneme mustn't be a semi-vowel
  (type 'speech-sampling rate sampling rate-assertion) ; find Fs assertion
  (let sampling rate (value sampling rate-assertion)) ; extract Fs
  (let stop-start (start stop)) ; starting epoch of $stop$
  (let stop-end (end stop)) ; ending epoch of $stop$
  ; find the nominal vot for this stop context
  (let avrg ms (vot mean s stop vowel stop 1000.0))
  ; locals representing the fraction of release that is frication
  (let fric fraction (frication fraction stop))
  ; and the typical standard deviation in $stop$'s VOT.
  (let deviation-ms (stop-deviation stop))
  PREMISE ODDS
  (lambda (stop left 2nd-left right sampling-rate deviation-ms avrg-ms)
    (min (odds 2nd-left) (odds left) (odds stop) (odds right)
        (let* ((dur (// ($length (domain stop))
                        .001 sampling-rate)))     ; dur in ms
          ; determine the binding confidence based on how typical
          ; this stop's VOT is, and the confidence in the stop context.
          (odds-gaussian (- dur avrg-ms) (* 2 deviation-ms) 5))))
  ACTIONS
  (let* ((dur ($length (domain stop))) ; stop release duration (in samples)
         (fric dur (round (* dur fric fraction)))  ; duration of frication
         ; Build an epoch to represent the boundary between frication and
         ; aspiration.  Uncertainty is given by the uncertainty of VOT.
         (epoch (epoch:shift stop-start fric dur
                  :standard-deviation (* deviation-ms .001 sampling rate))))
    ; assert marks for the fricative and aspirative portions or the release
    (assert (phonetic frication stop start epoch) .8 .4)
    (assert (phonetic aspiration epoch stop end) .8 .4))))
```

Figure 3.9 Voicing from Stop Timing

/s/<unvoiced closure><unvoiced stop release><not a semivowel>

The variable *avrg−ms* is assigned to the expected duration of the stop release. *fric−fraction* is assigned to the proportion of that release that is expected to be fricated, and *deviation−ms* is assigned to the expected standard deviation of the stop release. All three numbers are properties associated with each stop type, based on the phonetic context. This information comes from a study of stop durations[18].

In this rule the premise-odds form uses the statistics of stop duration and the confidence in the phoneme marks to assign confidence to the bindings created when the rule fires. As was mentioned earlier, the confidence in the binding in turn determines the support for the phonetic-frication and phonetic-aspiration assertions made in the actions of the rule.

A new epoch[8] must be created to represent the frication/aspiration transition during the release, and the rule provides information regarding the certainty of the position of that epoch. This is the purpose of *deviation −ms* in the ACTIONS of the rule. The form (*epoch :shift ...*) creates a new epoch shifted from the beginning of the stop (*stop −start*) by the duration of frication (*fric −dur*). The *:standard −deviation* argument to this function means the resulting epoch will have a larger standard-deviation than the epoch it was derived from and the amount of increased uncertainty is given by the *deviation −ms* expression in the ACTIONS.

Silent Gaps

Inferring the existence of gaps from the end of the phonetic transcript to the end of the utterance is accomplished with the rule shown in figure 3.10. This rule says "Find an utterance, and find a word that is also at the end of a phrase. Make a gap running from the end of the word to a new epoch created at the end of the utterance.". The reason for different versions of the procedure *end* in the conditions is that the utterance has no epochs at its start and end. This is also the reason for creating a new epoch at the location where the utterance ends (the standard deviation of the ending epoch was chosen as an intuitively reasonable value).

[8] "Epoch" is our term for the object that stands for the boundary between marks. Epochs are discussed in more detail below.

```
(defrule <phrase-end gap>
  CONDITIONS
  (type 'word-mark word)          ; a transcript word mark
  (type 'utterance utt)           ; the waveform
  (type 'phrase-mark phr)         ; the transcript phrase mark
  (end of phr word)               ; $word$ ends at phrase end.
  (let start (end word))          ; the ending epoch of $word$
  (let end ($end utt))            ; $utt$ is not bounded by epochs.
                                  ; Therefore, this is just a number.
  (< start end)                   ; Smart '<' fcn can compare the epoch
                                  ; $start$ with the number $end$
  PREMISE-ODDS
  ; binding confidence is given by the confidence in $word$ and $utt$
  (lambda (word utt) (min (odds word) (odds utt)))
  ACTIONS
  ; Since there was no epoch at the end of $utt$, we must make one.
  (assert (gap start (make simple-epoch :mean end :sd 100 :support utt))
          .9 .4))
```

Figure 3.10 Finding Gaps

This particular rule presents an interesting use of the support factors in the (*assert* ...) clause. The values of .9 and .4 say "This rule is very likely to detect gaps, but it also has a high false alarm probability". The likelihood of detecting gaps stems from the fact that the phoneticians are not likely to put phoneme-marks past the end of the sentence. The false alarm probability is high because even though there are no phoneme-marks there, there may be residual speech in the waveform (coughing, comments from the experimenter, etc.).

Location of Stop Bursts

It is helpful to determine where stop bursts occur because their brief high intensity can cause inappropriate voiced support from modules that determine voicing from power. In this situation, the retraction of support for the voiced conclusion caused by the existence of a burst assertion counteracts the power indicators.

The reasons for numerically locating stop bursts were:

- The estimates of the end of the burst from information about the phonetic environment of the stop were highly variable.
- This was an interesting opportunity to demonstrate one technique of combining symbolic and numerical processing. Namely, the use of symbolic information to select the area of application of a signal processing algorithm.

The rule which accomplishes this task is a simple one because most of the effort is contained in the signal processing procedure. As can be seen from figure 3.11, the rule looks for the utterance and an unvoiced stop release phoneme mark then calls the function *scan−for−burst* which actually makes the new assertions. This procedure finds the peak in power interior to the stop and works its way outward looking for a drop in power. Such a simple algorithm would be impractical as a purely numerical means of locating bursts, but in the tighter context provided by a stop phoneme mark it is effective.

Notice that the local variables *start* and *end* are not in fact used in either the ACTIONS or PREMISE-ODDS of the rule. This is an example of how local variables signify binding dependency. If those epochs were changed for some reason, the old

```
(defrule <numeric burst0>
  CONDITIONS
  (type 'p mark x)                    ; $x$ is a phoneme-mark
  (type 'utterance utt)               ; $utt$ is the waveform
  (isa ':unvoiced stop-release x)     ; $x$ must be a stop release
  (let start (start x))
  (let end (end x))
  PREMISE ODDS
  ACTIONS
  (scan-for-burst utt x))
```

Figure 3.11 Rule to find stop bursts

binding would become invalid and a new one would be created. Invalidating the old binding would retract any previous results from the procedure *scan −for −burst* . Subsequently, when the rule was fired again (due to the new binding), *scan −for −burst* would be reinvoked over the new region delimited by *start* and *end* , potentially yielding different results.

Also note that there is no PREMISE-ODDS form. This is because the support for any numeric-burst marks that are generated by *scan −for −burst* is determined by that procedure and not from any of the assertions that triggered this rule.

Voicing from Sonorant Power

Measuring the amount of low-frequency (sonorant) power (0-600hz) is one way of determining voicing. Unvoiced excitation of the vocal tract (e.g. frication) has mostly high frequency content, whereas glottal vibration always generates substantial low-frequency power. The rule for this task (figure 3.12) is like the one for bursts in that it runs a numerical procedure. In this case, there is no domain qualification by the symbolic information and the procedure is free to make numeric-voiced conclusions wherever sonorant power is high.

```
(defrule <numeric voiced1>
  CONDITIONS
  (type 'utterance utt)          ; $utt$ is the waveform
  ; $pwr$ is assertion concerning the maximum sonorant power
  (type 'max sonorant power pwr)
  ; $pwr value$ is the numerical  value of that assertion.
  (let pwr value (value pwr))
  PREMISE ODDS
  ACTIONS
  (scan for voicing (sonorant log power utt) pwr value))
```

Figure 3.12 Voicing from Sonorant Power

As with the previous rule, the odds of the created assertions are determined in the scanning process, not by condition variables. Thus there is no need for a PREMISE-ODDS form since the binding's confidence is irrelevant. This reflects two different purposes of variables in rules. One is the use of variables that have a direct bearing on the confidence of results, the other is the use of variables as a mechanism of controlling the behavior of the system (as in this case).

Voicing from Broadband Power

This rule is essentially equivalent to the previous rule, except that the scanning process looks for regions of the waveform that have power within a few dB of utterance minimum. The one thing that distinguishes this rule from the previous (as seen in figure 3.13) is the requirement of a *high −snr* assertion in the CONDITIONS. When experiments were undertaken with noisy speech, it became apparent that the algorithm used by this rule was ineffective if the speech was noisy. The *high −snr* is made by another module if the difference between maximum and minimum power is found to be greater than 40db.

```
(defrule <numeric silence1>
  CONDITIONS
  (type 'utterance utt)          ; the waveform
  (type 'min power pwr)          ; the minimum power assertion
  ; the waveform has a high signal to noise ratio
  (type 'high-snr snr)
  PREMISE ODDS
  ACTIONS
  (scan-for-silence (broadband-log-power utt) (value pwr)))
```

Figure 3.13 Silence from Broadband Power

Voicing from Similarity

This method of determining voicing was built into the system in a different way than the previous. Rather than running a scanning procedure and producing numeric-voiced assertions, this rule (figure 3.14) uses the time varying similarity of the waveform to establish the confidence in the binding and supports the final voiced result v directly.

The decision to implement this idea in this fashion was partly due to time limitation. The earlier examples all produced results in the form of voicing assertions which in turn supported the final voicing assertion. This two step process meant that other parts of the system had access to the results of those modules. This more direct implementation saved the extra step of producing numeric-voiced assertions at the expense of rendering the results invisible to the rest of the system.

```
(defrule <similarity-voiced support>
  CONDITIONS
  (type 'utterance utt)          ; the waveform
  (type 'voiced v)               ; the unique assertion <voiced>
  PREMISE ODDS
  (lambda (utt)
    ;; low sonorant power makes this measure invalid
    (seq lower
      (seq clip (seq exponentiate
                  (seq scale (sonorant log power utt) .05
                             (+ 20 (seq min (sonorant-log-power utt))))
                1.0 10.0)
       1.0 10.0)
      ;; use a lpf on the utt to eliminate uncorrelated power.
      ;; consider 2.0 similarity as warranting an odds of 1/1 on the premise.
      (seq clip (seq scale (local-similarity (gr-pd:gr-lpf utt)) 1.0 1.0)
                .1 100.0)))
  ACTIONS
  (provide support v .8 .01))
```

Figure 3.14 Voicing from Similarity

Another problem with this implementation of the similarity rule is that the epochs that would have been created for the numeric-voiced assertions were not made. So any improvement in accuracy that might have come from merging those epochs into the rest of the epochs of the system has been lost.

3.5. Determining F0

F0 is determined through the interaction of the following ideas: numerical estimation of the spacing between similar places in the waveform, f0 prediction from phoneme identity and speaker sex/age, f0 prediction from sex/age alone, f0 derivative prediction from phonetic environment and stress, and f0 range estimation based on preliminary (presumably trustworthy) estimates. This organization is depicted schematically in figure 3.15.

3.5.1. Concepts

Waveform Similarity

This part of the PDA corresponds to the core of conventional pitch detectors, the part which estimates f0 or period using numerical techniques. The basic principle is that periodic glottal pulses exciting a slowly varying vocal tract lead to periodicity in the acoustic output that can be used to determine f0.

F0 from Phonetic Identity

Using information available from speech research[9] the PDA can predict f0 from the identity of vowels and dipthongs and the sex/age of the speaker.

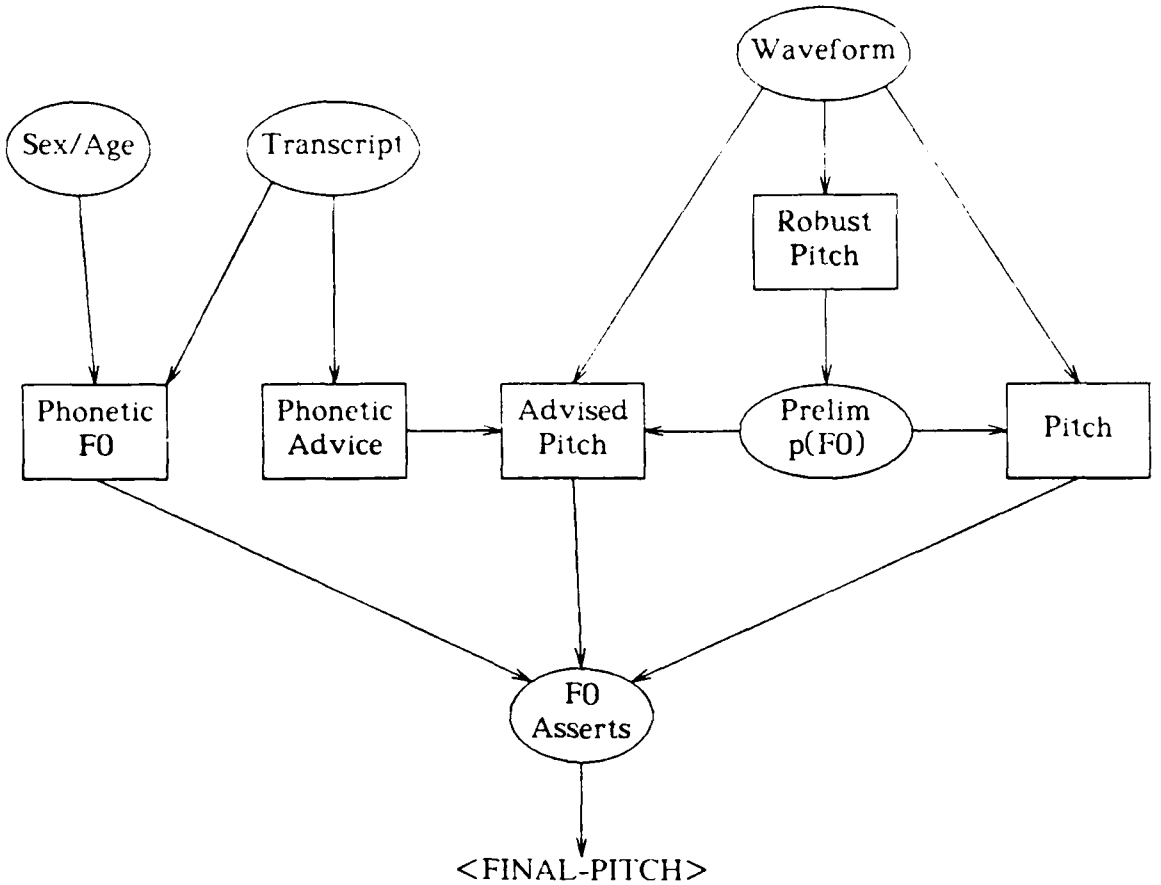


Figure 3.15 Determining F0

F0 from Sex/Age

With other phoneme classes, for which no f0 bias information is available, a prediction can still be made by using the data for a central vowel like /a/ and specifying a greater uncertainty in the estimate than in the phoneme specific case.

F0 Derivative from Phonetic Context

The effects of consonantal context on the f0 trajectory during vowels was investigated by Lea[13]. This information is used by the PDA to advise the numerical pitch

detector about the likeliest pitch trajectory at the beginning of certain vowels.

F0 Range from Preliminary Numerical Results

The pitch range within a single sentence is rarely greater than one octave, frequently less. In part, this range is being estimated by the phonetic identity and phonetic sex/age information above. However, by collecting a set of estimates of f0 that are felt to be reliable it is also possible to estimate the range numerically.

3.5.2. Examples of Implementation

Waveform Similarity

The numerical period estimation method used by the PDA is based on finding similar positions in the waveform. The technical details of the algorithm are discussed in Chapter 4, but in simple terms the algorithm windows the utterance at a location where the f0 estimate is to be made and compares that waveshape with nearby regions of the utterance that have also been windowed. Spacings between the windows that lead to similar waveshapes become candidates for estimating the period of the speech at that location. Figure 3.16 shows the similarity as a function of spacing in the middle of the vowel sound /i/. This algorithm is typical of time domain algorithms that are based on correlation measures in that it has a certain amount of background "clutter" between peaks and that there are peaks at multiples of the "true" period in both directions.

The rule responsible for computing f0 numerically is shown in figure 3.17. The CONDITIONS extract the utterance from the known assertions and find *apriori* $-f$ 0 which is a function that specifies the odds of a given f0 value. This is the result of the "f0 range from preliminary numerical results" module and is discussed below. There

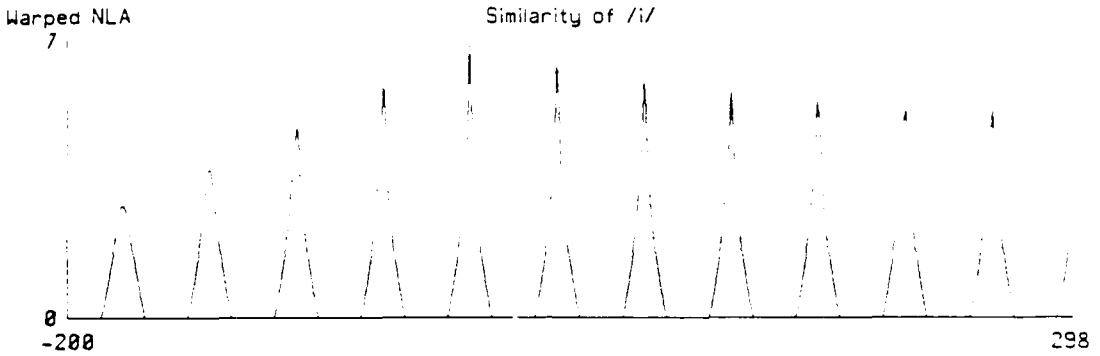


Figure 3.16 Typical Speech Similarity for /i/

```
(defrule <pd-on-remaining-support>
  CONDITIONS
  (type 'utterance utt)          ; the waveform
  (type 'apriori-f0 apriori-f0-assertion)       ; a preliminary f0 assertion
  (let apriori-f0 (value apriori-f0-assertion)) ; and its value
  (type 'do-final-pd switch)              ; operator control switch
  PREMISE-ODDS          ; confidence is determined numerically
  ACTIONS
  (let* ((phonemes (get-from-km 'p-mark))       ; all phonemes
         ; regions that have not yet been processed for similarity
         (support (support-without-sim pitch (apply '$cover phonemes))))
    (loop for ivl being the intervals of support do
      (format t "~&final pd on ivl ~a" ivl)
      (assert (sim-pitch-assertions (gr pd:gr-lpf utt) ivl 0.00
                nil .02 apriori-f0)))))
```

Figure 3.17 F0 from Similarity

is no need for a PREMISE-ODDS because the certainty information for the resulting f0 assertions are solely determined from the numerical measurements. A low-pass filtered version of the utterance together with this *apriori* $-f$ 0 function and some control parameters are used to invoke the numerical pitch detector.

The parameters passed to the procedure *sim* $-$*pitch* $-$*assertions* are (from left to right): the interval over which to detect f0 (*ivl*), the expected period change per cycle

(0.0), the "preferred" look direction (*nil*), and the range of possible period change per period (.02). In this particular rule, most of these control parameters are set to a neutral value (for example, no period change is expected and there is no preferred look direction). The other rules which invoke this procedure do supply "advice" in the form of values to these parameters, advice which depends on the phonetic context of the region being analyzed. The choice of .02 for the period "jitter" was selected based on speech research[26], and the results of a brief jitter experiment that we conducted. The details on the influence of these parameters on the pitch detection algorithm are presented in Chapter 4.

The variable *support* in the ACTIONS stands for the set of intervals that have not yet been numerically tested for pitch. Determining the regions that have already been analyzed is necessary because the implementation of the idea "f0 derivative from phonetic context" involves running the same numerical algorithm in a few intervals with advice about what to expect, and two sets of results from the same numerical algorithm in the same location of the utterance would conflict with independence assumptions made in the combination of such results. Thus this rule must restrict its actions to those intervals that have not already been numerically tested by this detector.

The existance of the switch *do−final−pd* points out a lack of complex control in this system. After the PDA has derived all it can from the phonetic input and waveform (including numerical pitch analysis in regions where phonetic advice is available), it will cease making assertions (without ever having run the previous rule). At this point, the operator adds the single assertion *do−final−pd* , triggering the above rule and performing numerical pitch analysis on the remaining regions.

F0 from Phonetic Identity

As was discussed in Chapter 2, tables are available that specify f0 given the phoneme and the speaker sex/age (either MALE, FEMALE or CHILD)[9]. In PDA, this information is used to generate f0 probability contours over the phonemes for which it is available. These contours have their mean given by the value in the table for that phoneme and speaker, the standard deviation is chosen as 30 Hz, and the odds-factor is given by the odds-factor of the phoneme in question.

The standard deviation was chosen as a balance between having some useful effect and encompassing the natural variations of different speakers. While the odds-factor is dependent on the odds-factor attributed to the phoneme, no use was made in this system of the ability to give individual phonemes different odds factors. Instead, they were all fixed with a nominal odds-factor of 4.0.

One of the two rules that stem from this idea is shown in figure 3.18. The other

```
(defrule <phonetic pitch from vowels>
  CONDITIONS
  (type 'p-mark pm)                          ; a phoneme-mark
  (isa ':vowel pm)                           ; a vowel
  (type 'sex sex-assertion)                  ; the sex assertion
  (let sex (value sex-assertion))            ; its value
  PREMISE-ODDS                               ; confidence determined numerically
  ACTIONS
  ; select the mean f0 from the sex/age
  (let ((f0 (inherit pm (selectq sex (:male :p-b-men)
                                     (:female :p-b-women)
                                     (:child :p-b-children)))))
    ; mean: f0, standard-deviation: 30hz, odds from the phoneme mark
    (assert (pitch-assertion #'(lambda (ignore) f0) #'(lambda (ignore) 30)
              (fcn-taper over-ivl
                (lambda (ignore) (send pm :odds-factor)) pm)
              (domain pm)))))
```

Figure 3.18 F0 from Phoneme Identity

performs the same task for dipthongs. The CONDITIONS require a vowel phoneme mark and the sex/age of the speaker. The PREMISE-ODDS are not used. The ACTIONS first look up the proper f0 value based on the sex/age using the (*inherit ...*) form. Then a pitch assertion is made using that f0, a nominal standard deviation of 30hz, the same domain as the phoneme mark, and a confidence equal to that of the phoneme mark (tapered to zero outside the domain of the phoneme mark).

In the dipthong form, the one difference is that the f0 value slides from the value attributed to the configuration at the start of the dipthong to that attributed to the end.

F0 from Sex/Age Alone

For regions of the utterance where the above information is not available, a more coarse f0 estimate is made based on the sex/age alone (see figure 3.19). In this case, the mean f0 comes from that for the neutral vowel /a/ for the speaker type in question. The standard deviation of 60 hz is broader than the above because this estimate is less well constrained. Finally, the odds-factor is chosen as a fixed 5/1 odds. In figure 3.20 we see the f0 information contributed by the above two ideas. The plot shows time extending from left to right, f0 extending from bottom to top, and probability density is shown by the darkness of the picture. A vertical slice through this surface would plot the probability density for f0 at that temporal location.

The relatively narrow densities (narrower vertically) are those contributed by the information that is phoneme specific. The wider density is the more general contribution for the remaining regions of the utterance.

```
(defrule <phonetic f0 on remaining support>
  CONDITIONS
  (type 'utterance utt)                    ; the waveform
  (type 'do final phonetic f0 switch)      ; operator switch
  (type 'sex sex assertion)                ; the sex assertion
  (let sex (value sex assertion))          ; its value
  PREMISE ODDS
  ACTIONS
  (let* ((phonemes (get from km 'p mark))
         ;; the support yet to be covered by pitch assertions
         (support (support without phonetic pitch (apply '$cover phonemes)))
         (f0 (inherit 'phoneme:<aa> (selectq sex
                                      (:male :p b men)
                                      (:female :p b women)
                                      (:child :p b children)))))
    ; iterate over the remaining intervals
    ; mean: f0 from /a/, standard deviation: 60, odds factor: 5
    (loop for ivl being the intervals of support do
      (format t "~&final phonetic f0 on ~a" ivl)
      (assert (pitch assertion (lambda (ignore) f0)
                               (lambda (ignore) 60)
                               (lambda (ignore) 5.0)
                               ivl)))))
```

Figure 3.19 F0 from Sex and Age

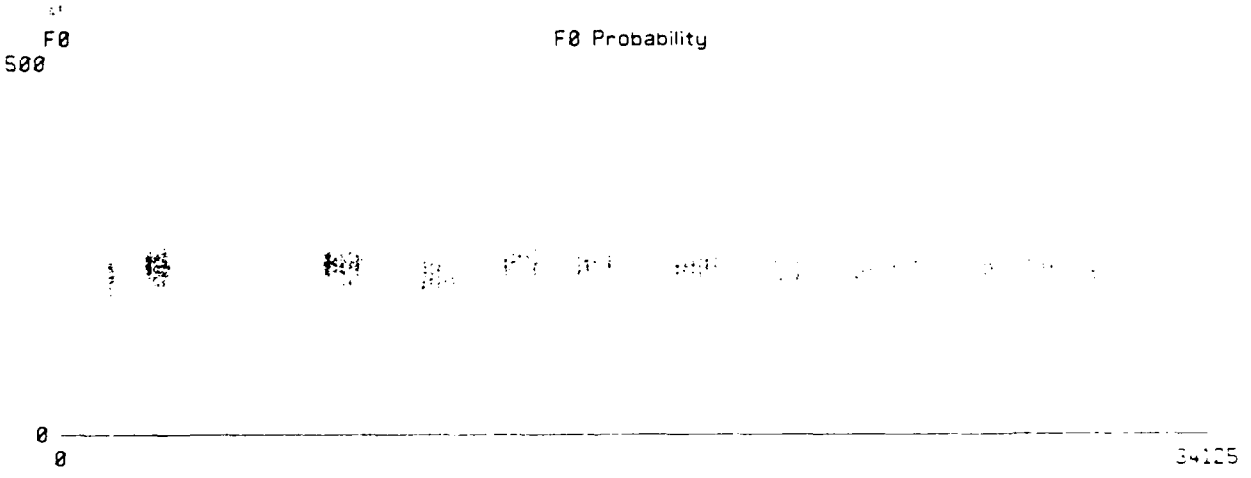
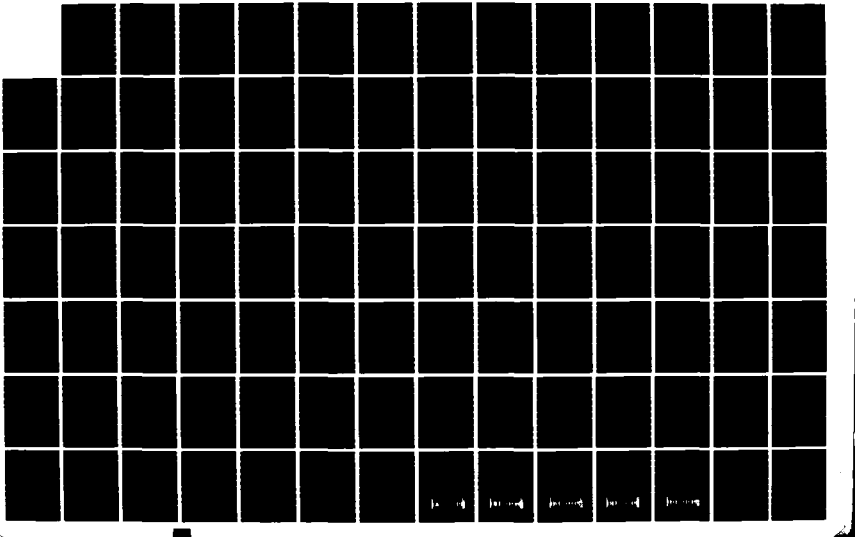


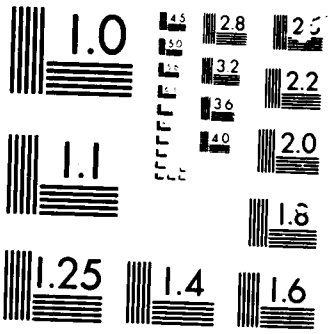
Figure 3.20 F0 from Phonemes

F0 Derivative from Phonetic Context

AD-A169 069 KNOWLEDGE-BASED PITCH DETECTION(U) MASSACHUSETTS INST 2/3
OF TECH CAMBRIDGE RESEARCH LAB OF ELECTRONICS W P DOVE
JUN 86 TR-518 N00014-81-K-0742

UNCLASSIFIED F/G 17/2 NL




1.0
1.1
1.25
1.4
1.6
1.8
2.0
2.2
2.8
3.2
3.6
4.0

Information about speech behavior near consonant vowel boundaries takes two forms. First, for certain cases there is an expected f0 change at the boundary. Second, sometimes one expects periodicity to be more apparent on one side of the boundary than the other. In both cases, PDA uses this information to "advise" the numerical similarity detector in those regions.

To employ information about the change in f0 near consonant vowel boundaries the numerical waveform similarity measure has the ability to make use of estimates of "period moveout". As is described in more detail in Chapter 4, the similarity pitch detector first finds a nearby region of similarity then looks for similarity near double and triple that distance. When given advice about "period-moveout", those multiple period locations are adjusted to reflect the expected moveout. Also the confidence result, which is based on how closely the double and triple period peaks are to the nominal values, is influenced by such advice. The directional advice also affects the confidence result by establishing a penalty for similarity found in the non-preferred direction.

The rule which performs this task is shown in figure 3.21. The CONDITIONS of this rule seek an unvoiced obstruent phoneme in a strongly or moderately stressed syllable followed by a dipthong or a vowel. The utterance, *apriori* $-f$ 0 and speech sampling rate are also acquired in the CONDITIONS. Since the confidence of assertions will be determined numerically, no PREMISE-ODDS is required. The actions first establish *ivl* as the first .03 seconds of the vowel, then the similarity pitch detector is used over that interval. Notice in this case that the control parameters of the pitch detector which were defaulted in the previous example (figure 3.17) are being specified here. For example, the expected period moveout is "positive" time (since voicing will

```
(defrule <advised-pd-1>
  CONDITIONS
  (type 'utterance utt)
  (type 'p-mark c)
  (not-null (memq (send c :phoneme-name)
                  'phoneme:(<p> <t> <k> <f> <th> <s> <sh> <ch> <q>)))
  (type 'speech-sampling-rate sampling-rate-assertion)
  (let sampling-rate (value sampling-rate-assertion))
  (type 'apriori-f0 apriori-f0-assertion)
  (let apriori-f0 (value apriori-f0-assertion))
  (let v (right-neighbor c))
  (or (isa :vowel v) (isa :dipthong v) (inherit v :y))
  (or (eq (send v :stress) 1) (eq (send v :stress) 2))    ; stressed
  PREMISE-ODDS
  ACTIONS
  (let* ((ivl (interval ($start v)
                        (+ ($start v) (round (* sampling-rate .03))))))
    (assert (sim-pitch-assertions (gr-pd:gr-lpf utt) ivl
                                  .01 :pos .02 apriori-f0))))
```

Figure 3.21 F0 Advice from Phonetic Context

grow stronger to the right).

F0 Range from Preliminary Numerical Results

The *apriori* $-f$ 0 assertion is generated through the actions of the two rules shown in figure 3.22 and figure 3.23. The first simply generates the preliminary-pitch-assertions using the low passed utterance. The second takes the histogram of those preliminary estimates that are very confident ("majority range") and establishes

```
(defrule <preliminary-pd>
  CONDITIONS
  (type 'utterance utt)          ; the waveform
  PREMISE ODDS
  ACTIONS
  (assert (preliminary-pitch-assertions (gr-pd:gr-lpf utt) (domain utt))))
```

Figure 3.22 Preliminary Pitch

```
(defrule <numeric pitch range>
  CONDITIONS
  (type 'preliminary pitch assertions prelim pd)
  (type 'final-pitch final pitch)
  PREMISE-ODDS
  ACTIONS
  (let ((range (send prelim pd :range)))
    (cond ((finite-interval-p range)
           (let* ((majority-range (send prelim pd :majority range .95))
                  (left ($end majority-range))
                  (right (+ left ($length majority-range))))
             (cond ((or (null-interval p majority-range) (< right 20))
                    (assert (apriori-f0 (lambda (ignore) 1.0))))
                   (t
                    ;; since the speech may be glottalized we place no
                    ;; limit on the maximum f0, only on the minimum f0.
                    (assert (apriori-f0
                             (lambda (f0)
                               (cond ((< f0 ($end majority-range)) 1.0)
                                     (t (kbsp:interpolate left f0 right
                                          1.0 .1 :clip)))))))))))
          (t (assert (apriori-f0 (lambda (ignore) 1.0)))))))
```

Figure 3.23 A priori F0

an "upper maximum" f0 by extending the maximum observed f0 ("lower maximum") by the spread of the observed f0 values. The *apriori−f 0* function that is asserted will penalize f0 values above the "lower maximum" linearly with f0 up to 10/1 odds against at the upper maximum. Note the tests in the actions that cause the *apriori−f 0* assertion to be simply a constant 1.0 (no effect). This occurs if there were no f0 estimates from the *preliminary−pitch−assertions* that were sufficiently certain, or if the number that passed were spread less than 20hz. These events can occur if there is sufficient noise that few "confident" numerical estimates occur. In this case, the rule becomes disabled.

3.6. Epochs

3.6.1. Concept

One definition of the word epoch is "a memorable event". This is the meaning intended by its use here. Some systems that attack problems related to time (e.g. SIAP and HEARSAY) represent the position of objects with numbers. For example, numbers are used to signify the time or frame index of the start of an object like a word (in HEARSAY) or a spectral line (in SIAP). In PDA we took an alternative approach of providing a complex data structure to represent time points. This decision was motivated by the complexity surrounding the manipulation of such events in this problem.

The significance of events

Speech is analyzed as a concatenation of objects. Words, syllables, phonemes, segments all break up speech into concatenated entities. To employ the symbolic knowledge available to it, PDA needs to know the locations of these objects. For example, it is known that a pitch fall can be associated with unvoiced-consonant/vowel boundaries. In order to position such an inflection, it is necessary to locate the boundary precisely.

The boundaries between these objects are what the PDA considers "significant events". In the course of analyzing an utterance, several pieces of (relatively) independent information may be given about the position of some particular boundary. For example, to locate an uv-consonant/vowel boundary there will be information in both the input transcript and positions of threshold crossing generated by the numerical power measuring procedures. To do the best possible job of estimating the positions of these events, there must be a means to combine the information from all these sources. Combining position estimates is one purpose of the Epoch system.

Alignment of Symbols and Waveforms

Another important use of the Epoch system is the precise alignment of numerical and symbolic information about the utterance. The PDA uses both numerically derived and symbolically derived information to guide the operation of some numerical procedures. For example, if the numerical period estimator is to be advised about a likely f0 fall after an uv-consonant/vowel boundary, then there must be a single statement about where it is expected to occur. Thus, the position information derived from separate numerical and symbolic sources must be combined before it can be used to guide the period estimator.

The PDA aligns the phonetic transcript and numerical energy measurements with a single Epoch system. This means that an epoch hypothesized by a symbolic module and an epoch hypothesized by a numerical module will be "merged" into a single epoch if the system concludes that they refer to the same underlying event. This "merger" produces a single "composite" epoch that combines the statistical information from both of the constituent "simple" epochs for better accuracy, and also serves to logically connect (in a network of epochs and objects) those objects that either of the two epochs were previously associated with.

All of the above points can be seen in figure 3.24. The small boxes represent the epochs. The width of the box depicts the uncertainty in the position of the event that the epoch stands for. Identical epochs that are exactly above one another are in fact the same epoch redrawn at a different level.

The words, syllables and phonemes are connected with one another because they came from a single set of simple epochs (generated by the entry of the phonetic transcript). Likewise the interval marks starting with 'p' (ps=phonetic silence,

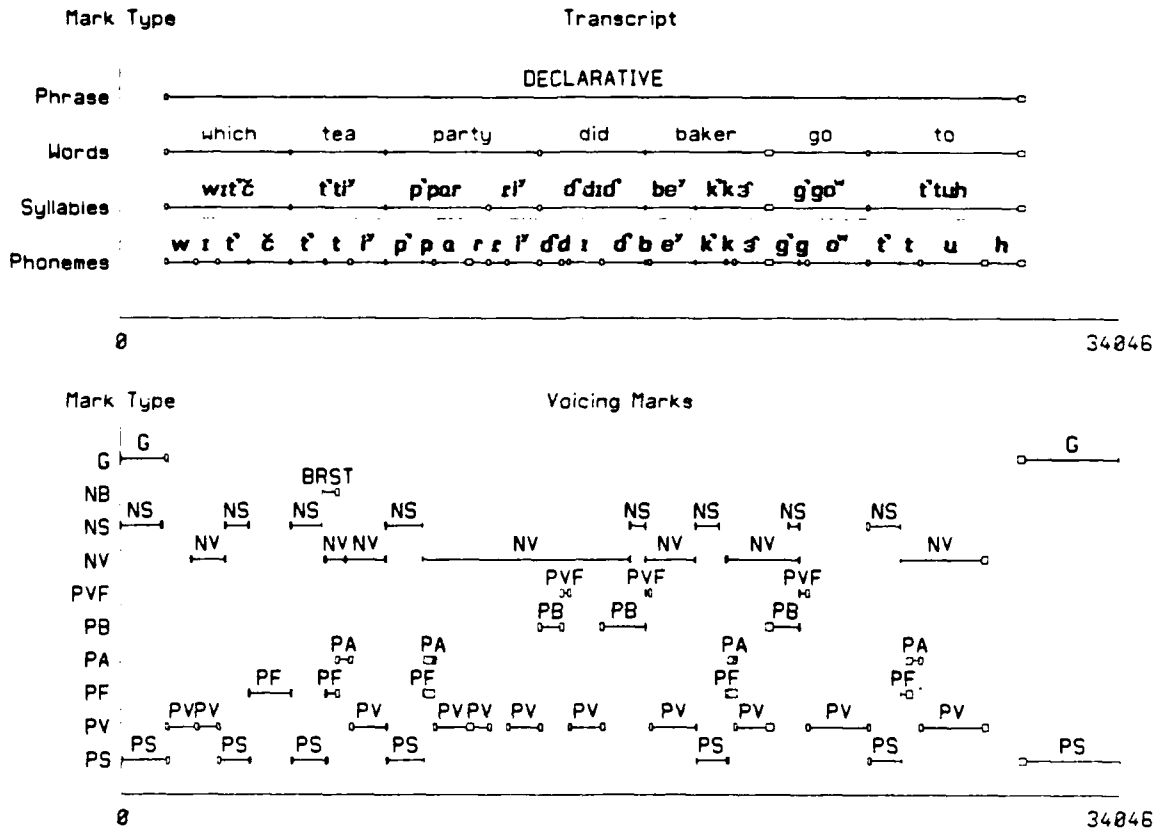


Figure 3.24 An Example of Epochs

pv=phonetic voiced) were derived from the phoneme marks, so they too share the simple epochs that were created by the transcription. However, the 'n' marks (numerical silence, numerical voiced) were derived from power measures made on the waveform and these procedures created their own set of epochs to delimit the numerically computed regions. It is because of the Epoch system that the numerical and phonetic interval marks are tied together.

Computing temporal position

Pitch detection includes both voicing and period determination. The correct time alignment of these two properties with the speech is essential. Since there are inaccuracies in the input transcript and in numerical measurements of the waveform, it is useful for PDA to combine its sources of temporal estimates to determine event positions more accurately.

This combination is accomplished through a largely numerical process derived from a few theoretical assumptions. Each epoch includes statistical information that models the position of the event it represents as a Gaussian distribution (by specifying mean and standard-deviation). When two or more epochs are "sufficiently" close and are compatible, they are assumed to stem from the same underlying event. They are further assumed to be independent estimates. These assumptions lead to a formula for determining the mean and variance of the "composite" estimate. A more detailed description of the implementation is provided in Chapter 4.

The assumptions

The assumptions of independence and Gaussian statistics were made largely for convenience. It seemed appropriate to use a unimodal distribution for the statistics of each event estimate. Smaller errors were more likely than large ones (so it shouldn't be uniform). The Gaussian shape is intuitively familiar to most people with a mathematical background, so providing accuracy assessments in terms of Gaussian standard deviation was convenient[9]. Estimates were also assumed to be independent because if there was dependence there was no readily available way to determine it,

[9] The operator must provide in the transcript an indication of the inaccuracy of each position estimate so the system can properly merge them with the temporal estimates that are automatically generated.

and a dependence assumption would require some sort of specification.

These are certainly powerful assumptions to make, but they seemed reasonable in this context and they allowed us to derive the mechanism of combining the statistical estimates rather than picking an ad-hoc procedure. Further, there did not seem to be another preferable set of assumptions given the information we had about this problem. While the independent Gaussian assumptions we made are probably wrong in many cases, any other set of assumptions would be as well. If the performance of the system were critically dependent on the precision of specifying the statistics of these estimates, then building a workable system would probably have been impossible.

Sufficiently close

Each epoch provides a probability distribution for the position of the underlying event it represents. Assuming that a number of such epochs come from a single event, one can view the probability density as specifying the position of the epoch given knowledge about the true event position. Thus if many epochs pertain to a single event, they form a cluster about that event and the system has (estimates) of the statistics of that cluster that can be used to evaluate it.

Specifically, if the epoch cluster is viewed as a point in n-dimensional space near the point that corresponds to the true event, it is possible for the system to assess the chance that the epoch point would be further from the event point than it is. It is this evaluation that the Epoch system uses to decide if two epochs are sufficiently close (the details a covered in Chapter 4).

Consistency of Merges

Another requirement for epoch merging is consistency. It became apparent in experiments that the criteria of "closeness" was insufficient. The most prominent example was when the starting and ending epochs of a measured burst were merged. This occurred because they were both uncertain enough to have plausibly been from the same event. However, in this case it was logically inconsistent to merge them since they were known to be separated by the burst.

To deal with logical information about merging, a filtering mechanism is provided in the merging process. In the PDA, the only filter used specifies that the starting and ending epochs of a given object may not be merged. However, clearly other criteria for merging could be considered. For example, in figure 3.25 one can see failures: the numerical burst (BRST) should not extend into the middle of phonetic silence region

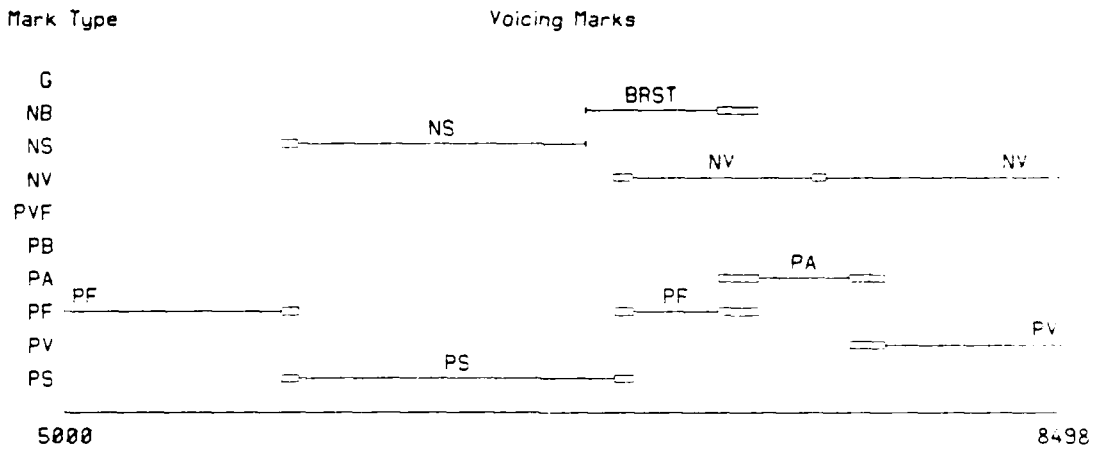


Figure 3.25 Bad Merging

(PS), and the numerical voiced region (NV) should not be merged with phonetic frication (PF). In the former case, the numerical burst represents sound and is therefore incompatible with phonetic silence. In the latter case, the NV should signify a voiced region; which cannot be the case during the phonetic frication implied by an unvoiced stop. These situations can and should be detected and corrected by the system. However, there was insufficient time in this project to deal with these issues. We merely wish to point out an important potential use of combined numerical and symbolic processing that should be investigated in future work.

Epochs as nodes

Epochs serve as the connections between objects. Adjacent phonemes are linked through the epoch that ends one and starts the other. Different levels of abstraction such as a word and its starting phoneme both start with the same epoch. Different concepts such as a numeric-voicing assertion and a word may start with the same epoch. Thus conditions in rules which express connectivity constraints may conveniently be determined through examination of the network of objects and epochs.

Epochs as Abstraction

Epochs combine various things related to events under one system.

- The determination of event positions by combining estimates.
- The representation of adjacency between objects via a network for efficiency.
- The determination of connectivity in that network through a combination of numerical closeness and logical consistency tests on the epochs of the objects.

By representing events explicitly in this way, other parts of the system can make use of epochs in one abstract way without being concerned with their other uses. For

example the rule interpreter can use the connectivity between epochs and objects to determine the left neighbor of some phoneme without being aware of the nature of epochs as numerical event estimates. This sort of abstraction would have been difficult to achieve if epochs were represented simply as numbers.

3.6.2. Related systems

Another system whose purpose is the representation of temporal information is given in[53]. This system allows the user to make statements about intervals in time in symbolic terms such as: "interval A begins before interval B", "interval D is inside interval F", etc. The system can then determine the truth of other statements about the intervals in question.

This system was intended for the description of time in the context of issues of causality. For example, in some expert systems[54] it is important to know the order of events to determine the meaning of certain observations. For example, in medical diagnosis there are numerous interacting systems within the body. The order of events (such as a rise in blood pressure versus a change in weight) may be needed to make the correct diagnosis.

The purpose of the Epoch system is different. It was not intended for determining causality since the relative relations of the objects in PDA's problem are largely clear. The Epoch system provides a mechanism for integrating multiple uncertain estimates to improve precision. Further, the Epoch system serves as the structural linkage for the temporal assertions of the PDA. Neither of these capabilities is part of Allen's system.

3.7. Knowledge Manager

3.7.1. Overview

The knowledge manager is a part of the PDA which stands between the rules and the asserted information of the system (figure 3.26). It runs the rule system, maintains the dependency network, maintains the confidence support network and computes confidence for scalar and distributed assertions.

3.7.2. Running the Rule System

The rule system is run in three phases: check changed assertions, check new assertions, fire rule bindings. The latter two tasks are typical for a rule system (though the PDA has a different control regimen than most current rule systems). The first task is unique to PDA and is something that contributes to the efficiency of the rule system. In most rule systems, rule actions may only assert objects. To implement change, the old object is removed and replaced. In the PDA, when rules are invoked they may create new objects or change old ones.

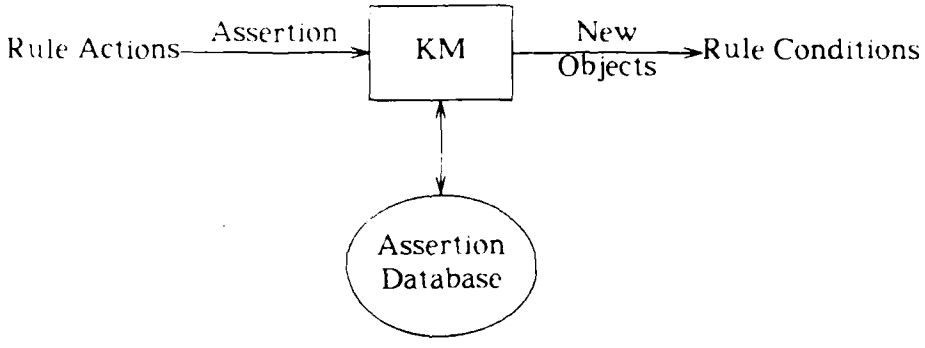


Figure 3.26 The Knowledge Manager

When changes are made, notices are passed along links between the various objects in the system informing dependents that the change occurred. This may immediately cause the dependent to respond. For example, when a word-mark learns that one of the syllables it covers has changed, the word-mark checks to see if its starting epoch and ending epoch match the starting epoch of the first of its covered syllables, and the ending epoch of the last. If not, then the word-mark updates its start (or end) and itself sends out a change notice.

Most objects are not sensitive to change notices. However, the knowledge manager is listed as a dependent of every assertion and it does respond to change notices. It keeps track of all changed objects during a particular rule firing session and subsequently checks the status of all rules with respect to these changed objects (as described below).

Another type of object that always responds to change is the **binding**. Recall that a binding stands for one invocation of a particular rule. Also recall that in the PDA, if the circumstances that justified that rule invocation go away, then the effects of that rule invocation are undone as well. When a binding learns that one of the objects that triggered its creation has changed it marks itself as "invalid". During the "check changed assertions" phase of rule system operation, the KM feeds all changed objects back through any rules that were interested in them. This can lead to three possible outcomes for any rule.

- An invalid binding may be made valid. If the rule finds that the changed object still satisfies the rule's conditions then the existing binding will be made valid and the actions it caused will remain in effect.
- An invalid binding may remain invalid. If after all changed objects have been checked, the rule responsible for the binding does not find that the conditions for that binding are now satisfied then the binding will be revoked and all of its effects undone.

- A new binding may be created. It may be that a changed object now satisfies a rule criterion that it previously did not. In this case a new binding will be created (though it will not perform any rule actions until the "firing" phase of rule system operation).

Once the old bindings and changed objects have been checked, the next step for the KM is the determination of which rules trigger (generate bindings). Some new bindings may have already been created during the checking of changed objects. The remaining new bindings are found by testing the rules against any new assertions.

When new assertions were made during the preceding rule-firing phase, the KM noted all rules that were potentially interested in those assertions. At this point, the KM sends the new assertions to those rules and gets back notification from any rule that was able to create new bindings.

Now the KM knows about all new bindings that are based on the previous rule firing phase. Also, any results that were invalidated by changes made during the previous rule firing phase have been undone. At this point, the KM uses the new bindings to invoke their rules leading to a new set of changes and new assertions.

The ability to handle change in the assertion database is novel in the PDA rule system. There are truth maintenance systems (TMS)[55] that can be used to build networks in which cells are either on, off, or unknown and are connected with "or" and "not" to model logical expressions. One of the uses for the dependency network in the PDA is similar to a TMS. When a binding is retracted and it notifies all its dependents (the assertions it made during rule firing) and they are also retracted, then the PDA is acting very much like a TMS.

However, there are several ways in which the PDA differs from a conventional TMS. Historically, TMS systems work with fixed networks of cells and update the

network in accordance with changes in the states of those cells imposed from without. The PDA differs from this since it is constantly changing the network as assertions are made (each assertions corresponding to a cell). In addition, the PDA uses this network not only for passing information about the logical state of a cell (i.e. has it been retracted), but also inform the dependents that a change has taken place in which they might be interested, a message that does not necessarily affect the informee. In this sense, the "cells" in the PDA have a more complex state than simply on, off or unknown and more complicated ways of changing that state. Still, the ideas behind TMS certainly had a strong influence on the PDA.

Existing rule systems do not work by having assertions each with a complex state that changes with time. They implement change by removal and replacement of the object in question. In a system where objects are simple and assertion is inexpensive this is a reasonable thing to do. However, in the PDA (and likely in similar KBSP systems) some assertions may take many minutes to create, and the alteration of the network structure to add and remove assertions can be expensive. Further, if consistancy is to be maintained when an assertion is removed, all dependent effects must be undone. This can involve great expense. Thus, the desire to automatically undo actions if the rule activation is no longer valid, coupled with actions that can be expensive to take (typical of many signal processing operations), leads to a design which explicitly handles change in objects rather than using removal and replacement.

The other unusual aspect aspect of the PDA brought out in this discussion is the simplistic control regimen. Existing rule systems create a conflict set of bindings and select a single member to fire. For example, the oldest or newest binding may be chosen or the binding with the largest number of condition clauses may be chosen. In

contrast, the PDA purposely lacks such a control mechanism. All bindings are collected then all bindings are fired. This choice was motivated by the belief that embedding knowledge in the control mechanism could lead to an obscure system that would be difficult to develop. Thus PDA lacks such a mechanism. The user relies on the consistency maintenance discussed above and explicit conditions in the rules to determine system behavior.

3.7.3. Network Maintenance

Objects in the PDA are related through two networks, one for dependency and one for probabilistic support. The first network is for absolute dependence (object "a" must exist for object "b" to exist) and the notification of change. For example, a numerical power estimate depends in this way on the waveform. If the waveform is retracted (killed) the power estimate must be retracted as well. The second network is for computing confidence. If support is removed for an assertion like voiced, the confidence in voiced may change, but the assertion is not removed. The odds factor just reverts to 1.0, and the odds to its a priori value.

The two participants in a dependency link are called the **user** and the **server**. Typically, each server provides something that its user needs. If the server changes, the user is informed in case the user needs to act on that change. If a server is killed, the user typically dies as well. All signals created by the system are made to "use" the signals from which they are constructed. In this way, if a signal is removed (say the algorithm for computing it is found to be incorrect) then all signals computed from it are removed also. Likewise, bindings are made to "use" all the objects that went into activating the rule. If any of those objects are killed then the binding dies. Also, if any of them are changed, the binding is notified and marks itself invalid (as

described earlier). The KM is also made to "use" all assertions so it too will be notified of object change and can check the changed objects against interested rules. Unlike bindings, the KM will not die if one of the objects it is using does. Finally, all assertions made during rule actions "use" the binding for that activation so if the binding dies, all assertions are removed.

The support links are established in two ways. During rule actions, new assertions may be "supported" by the binding if the confidence in that assertion stems directly from the objects matched in the conditions. The rules for phonetic voicing from phonetic identity are of this type (see figure 3.8). On the other hand, some assertions have their confidence determined by a numerical procedure. In this case no support is provided, and the default confidence is established when the assertion is created (see for example figure 3.12). The other means of creating support links is the explicit request. For example, all phonetic-voiced assertions ultimately lend (or remove) support from the unique assertion **<VOICED>**. This is handled by the rule shown in figure 3.27. The **provide-support** form causes the binding of this rule invocation to "support" the object of type **voiced** (which is the unique object **<VOICED>**). The binding is in turn "supported" by the objects that appear in the PREMISE-ODDS form

```
(defrule <pv voiced support>
  CONDITIONS
  (type 'phonetic voiced pv)
  (type 'voiced v)
  PREMISE-ODDS
  (lambda (pv) (ivl gate pv (odds pv)))
  ACTIONS
  (provide-support v .7 .3))
```

Figure 3.27 Phonetic Support for Final Voiced

(pv in this case). Notice that while the binding "uses" **<VOICED>**, it is not "supported" by it. If **<VOICED>** was retracted for some reason, this binding would have to be retracted, but the confidence in the binding does not depend on the confidence of **<VOICED>**, only on the confidence of the phonetic-voiced assertion found by the conditions.

In maintaining these networks, the KM must make the links between these objects, propagate notifications along **user** and **server** links, propagate changes in confidence along **support** links and recompute confidence when necessary.

3.7.4. Confidence Computation

When the support for an object is changed, the confidence in that object is usually affected. In terms of the nature of their confidence, there are three types of assertions. Symbolic-scalar, symbolic-sequence and numeric-sequence. A symbolic-scalar assertions makes a statement without reference to time index (e.g. sex/age=CHILD), a symbolic-sequence assumption refers to the time index (e.g. phonetic-voiced gives the confidence of glottal voicing as a function of index), and a numeric-sequence asserts a numerical value as a function of index (e.g. f0 versus index). The KM treats the first two types in a similar way and the last differently.

Symbolic assertions (sequential and scalar) are supported by bindings. The KM computes the odds-factor of such an assertion using a formula similar to that used in PROSPECTOR[56]. All the incoming support is combined somewhat like water flowing in (or out) along the support links. For symbolic-scalar assertions, the PDA resembles PROSPECTOR a great deal. For symbolic-sequence assertions (which are new to PDA), the odds-factor at each point in the domain of the assertion is computed

separately and kept in a confidence sequence for that assertion[10]. This allows it to make such assertions on intervals over which the confidence is varying. The PDA seems to be the first system to compute the confidence in distributed symbolic assertions in this way. Other systems which use such assertions (e.g. HEARSAY) employ a fixed confidence over their entire duration.

For the numeric-sequence assertions which represent estimates of F0 a different approach is used. The rules which give support to the final pitch assertion do so by directly linking the supporting pitch assertion with the final pitch as is shown in figure 3.28. The assertions which provide the support do so by specifying the parameters of a Gaussian density and a confidence at each index. At each sample in the final pitch, the KM takes the probability densities from the supporters and their confidences and makes a multi-modal composite density using conventional rules of probability (see Chapter 4 for details). Because it generates a multi-modal density it is capable of reflecting various possible choices for f0 with different peak amplitudes.

```
(defrule <phonetic support for final pitch>
  CONDITIONS
  (type 'final-pitch final-pitch)
  (type 'pitch-assertion assert)
  PREMISE ODDS
  ACTIONS
  (send assert :support final-pitch))
```

Figure 3.28 Support for Final Pitch

[10] Because of the efficiency of the signal processing system which the PDA uses, it is possible to take such an approach despite having in effect many millions of confidence computations to perform.

3.8. Rule System

This section discusses the concepts that motivated the design of the rule system. The responsibility of the rule system is the recognition of the conditions for activating rules. The rule system in the PDA supports the following features: networked objects, mutable objects, a dynamic rulebase, and test-directed rather than pattern-directed invocation.

3.8.1. Concepts

Networked Objects

Objects in PDA are all connected together. The rule system works with such a structure. Many rule systems (such as OPS5) do not. Their assertions have attribute values that are symbols. This contrast can be best understood by way of figure 3.29.

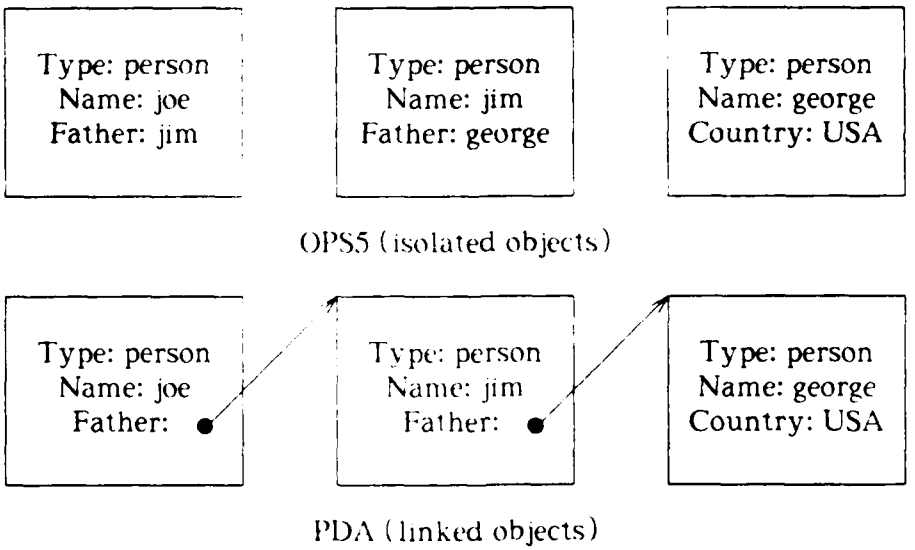


Figure 3.29 Networked Objects

In a rule system such as OPS5, asserted objects are represented by a data structure that records attributes and values for that object. In OPS5, these values can only be scalars or symbols. In terms of the figure, this means that in order to find out whether Joe had an ancestor from the USA, the rule system must take Joe's father's name (Jim) and test it against all other person's names for a match, then see if that person was from the USA. When that fails it repeats the process with Jim finding George. This expensive process can be avoided if the objects are connected as is shown for PDA. In this case, the function **inherits** can follow the pointers from son to father until the information is located.

The ability to directly search for such information can mean an improvement of the order of the number of assertions being searched or possibly powers of that order (depending on the rule conditions). This is discussed more fully in Chapter 4. In the PDA, the networked *representation* led to a substantial cost savings (as it would in many problems it was applied to).

Mutable Objects

Mutable objects in the rule system means that objects can be changed after being asserted. In the PDA, the primary source of object mutation is the Epoch system. As objects are linked by it, they change. For example, as a result of an epoch merger a word may become the neighbor of a voicing onset, and this may trigger a rule to fire.

A second posible source of change is the explicit modification of objects by rules. In one experimental rule, the voice onset time of stops (marked in the transcript) was compared to tables of expected onset times. If the marked onset time was too large and the stop was at the end of a word, the PDA would unlink the two words and insert a word pause automatically. Such changes have to be detected by other rules if

the system is to function properly.

Rule systems generally rely on the removal and replacement of an assertion in order to change it. In the PDA (and we feel in other KBSP tasks), such an approach to dealing with change would be too costly. The existence of dependencies between objects means that the removal of one object leads to many others being removed and substantial amounts of work being redone. Another problem with removal and replacement is that some objects (e.g. numerical pitch estimates) are very expensive to compute, so removal and replacement is very undesirable.

Since some rule systems have functional conditions, they could be used with networked objects. Why can't they handle mutable objects? The problem is in the way change is detected. These rule systems have a network built from the conditions of all the rules. They locate the rules to be fired by dropping new assertions into that network. Once an assertion has been fed to the network and thereby tested, there is no provision for detecting change in that object and testing it again. It must be removed and replaced to accomplish that.

Dynamic Rulebase

The PDA was intended to evaluate ideas for developing complex systems. One aspect of such development is the testing and debugging of new rules, and the ability to dynamically change the rulebase improves the efficiency of program development. In some rule systems, to alter a rule one must remove and replace all assertions. In the PDA, a complete restart would take a long time. By allowing rules to be added or removed at any time, the developer can experiment with a new rule in a matter of minutes.

Test-Directed Invocation

The PDA uses functional tests rather than patterns as the primary way of expressing conditions. Since there were numerical quantities to be manipulated in conditions, such tests were inevitable. Many of the non-numerical conditions extracted information about objects that was not directly present in the object itself and would therefore be difficult to implement with simple (efficient) patterns (see the previous comments about networked objects).

Finally, tests are functional abstractions in that they do not constrain the actual data structure of the assertions. Pattern-based rule systems supply you with a data structure that their pattern language supports. By avoiding such patterns we remove any structural constraint on the assertions in PDA and allow those aspects of the system to be programmed independently of the rule conditions.

3.9. Conclusions

This chapter has given a conceptual presentation of the PDA system. It discussed the program knowledge and how it was used and developed the ideas behind the major subsystems (Epoch, KM and rule system). The following chapter goes into detail on these systems, presenting the theoretical mathematics behind the numerical similarity estimator and the confidence computation. Chapter 4 also introduces the KBSP signal processing system and discusses how it was used in this project.

CHAPTER 4

System Details

This chapter describes the implementation details of the numerical pitch detector and other major components (the KBSP package, the Epoch system, the rule system, and the knowledge manager) that make up the PDA.

4.1. The KBSP Package

The Symbolics LISP machine on which the PDA was implemented supplies the programmer with a LISP language environment that includes a LISP oriented editor, a debugger that allows the examination, modification and resumption of all executing procedures, support for the creation and modification of data types to encourage object oriented programming, and a large system for controlling the graphic display.

This LISP language environment provides a powerful base on which to build applications, but it does not support specific applications. In particular, the LISP machine software contains no datatypes or procedures designed specifically for signal processing. The KBSP package extends the basic LISP machine software environment with datatypes, procedures and graphical displays that help the programmer develop and use signal processing algorithms.

The KBSP package was the joint effort of the author and Cory Myers who was working concurrently on his PhD thesis. The following text briefly describes the major features of the KBSP system. A much more detailed account is available in the 1986 PhD thesis of Cory Myers which presents extensions to the KBSP package that

allow signals to be manipulated in symbolic as well as numerical form.

4.1.1. The Precursor (SPL)

One of the first efforts to specify the desired features of a signal processing datatype was made by Gary Kopec[57]. Kopec's work was based on the principle that it would be easier to write signal processing programs if there was a basic datatype from which all signals were constructed. Designing a procedure to process signals would be simpler because the external interface presented by all signals would be identical. Developing large systems would be simpler because changes in the internal implementation of signals could be made without requiring changes in the procedures that used them. This would increase the chance that programs written at different times and by different people would be compatible.

There had to be a common functionality that all signal objects supplied, which was sufficient for any of the operations that might be performed on them. For example, Kopec pointed out that a signal representation should support non-sequential access since many signal processing procedures are described in that fashion (e.g. the FFT). He proposed two operations that a signal must support: (fetch <signal> index) and (domain <signal>) . The first is needed for retreiving the samples of a signal, and the second for identifying which indices are "defined" (attempts to access a signal outside its domain are program errors in his model of signals).

Kopec also proposed some other properties that a signal representation should have. He justified these properties by saying - "since mathematical signals have them, computer signal representations should have them as well". For example, he felt that signal objects, once created, should never change (since mathematical signals do not). He suggested that the closer the programmed representation of a signal was to the

mathematical concept, the better a signal processing tool it would be. To achieve this he mapped properties of mathematical signals to properties of computer signals.

The Signal Model

Kopec implemented this philosophy in a signal processing package he referred to as SPL. SPL modeled signals as objects whose domains were finite intervals (e.g. $\{0,\ldots,(N-1)\}$) and which supported the two operations (*fetch* signal index) and (*length* signal). It was an error to fetch a signal outside its domain. Thus SPL uses what might be called a "finite vector" model for signals. Signal objects (like mathematical vectors) are only defined over a finite range of indices starting with 0, and it is an error to combine (e.g. add) two signals of different lengths. First the shorter must be used to create a zero-padded version of the same length as the longer, then they can be added.

Signal Representation in SPL

Signals are represented by objects that contain two procedures, one for calculating the samples of the signal and one for calculating the length. Thus the basic task of "creating" a signal is one of "definition" not "computation". Rather than creating a Hamming window by computing all its elements and storing them in an array, the program defines the procedure for computing those elements. Note that such definitions may take place at compile time (as with conventional FORTRAN subroutines) and while the program is running (as procedures which create signals are invoked). The invocation of the procedure for computing the samples of a signal only takes place when some other procedure asks for the sample values of the signal (e.g. via (*fetch* signal 17)).

Signal Definition

If the definition of every Hamming window of a different length had to be written as a separate piece of code, signal processing would be a difficult chore. To simplify the definition of signals, SPL permits the user to define an entire class of signals with a single block of code (e.g. all Hamming windows are defined with one piece of code). For example, the signal class definition for Hamming windows creates two procedures that are generic versions of the fetch and domain procedure for a specific Hamming window, and a third procedure called a builder.

When a specific Hamming window is desired, the builder for the Hamming window class is invoked passing it the desired length. The builder creates an object to represent that specific Hamming window by customizing the generic Hamming fetch and length procedures for the specified length of this specific Hamming window.

4.1.2. The KBSP Package

The KBSP package used Kopec's work as a starting point. It includes the idea that signals should be defined rather than computed (i.e. a signal is more like a function than an array), and that the user should be able to define whole classes of signals at once. However, KBSP differs from SPL in some important respects.

The Signal Model

The signal model used by KBSP is different from the one used by SPL. Its distinctive features are as follows:

Domain
: In KBSP the domain of a signal is $(-\infty\ \infty)$.

Period
: All signals have a period (for most signals it is infinite). The combination of two periodic signals generally leads to a signal whose period is the least

common multiple of its constituents. Signals which are not periodic are said to have period=∞.

Support

The support of a signal is a range of indices outside of which the value is 0. Some signals (e.g. Hamming windows) have a finite support, others (e.g. real exponentials) have infinite support.

Benefits of the Model

- Because this model supports infinite domains, the class of representable signals includes common periodic signals and infinite support signals like exponentials, Gaussians and sinusoids.
- Infinite domains also means the preservation of temporal properties such as zero-phase, causality and temporal offset is straightforward. For example, the convolution of two zero-phase signals will still be zero-phase.
- All signals have the same index domain, so any signal may be combined element by element with any other. Thus constructions like (*seq-sum* (*Hamming* 31) (*Hamming* 43)) are well defined in KBSP.
- Infinite domains means the user of a signal need not be concerned with the support of a signal except for efficiency. For example, when adding two signals of unequal support the user is allowed to fetch samples from the shorter signal outside of its support without causing errors. This makes many procedures that process multiple signals simpler.

Side Effects of the Model

Deferred Evaluation

Choosing infinite domains makes deferred evaluation a requirement.

Interval Algebra

The computation of support is an important part of the definition of a signal. Some signals (such as correlations) have a support which is a complicated function of the supports of their arguments. To assist the programmer in manipulating supports, a datatype for representing them was created called "interval" and a set of procedures for manipulating intervals were defined. Operations such as union and intersection were provided as well as signal processing related operations such as shift, correlation and convolution of intervals. These procedures made use of explicit symbols for ∞ and −∞ to make the definition and manipulation of signals with infinite support convenient.

Fetching Intervals

One requirement for a signal representation is that it be computable at an arbitrary index. From a practical perspective, there is a computational cost associated with each access. Thus for efficiency reasons, it is useful to be

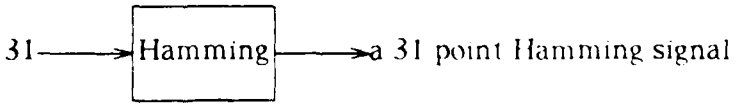
able to access many samples of a signal at once (typically a group of samples would be stored temporarily in a fast access data object like an array). Recent versions of Kopec's language allow the user the alternatives of fetching a single sample from a signal or all the samples. In the KBSP package it is impractical to fetch all the samples (since the domain is infinite). Instead, the user is allowed to fetch an arbitrary interval of samples (which are returned in an array). Computing an interval of samples of a sequence usually involves fetching intervals of samples from the sequences on which it is based. Determining which samples of those input sequences are required can be a difficult aspect of defining a signal (esp. for correlation). This is another use for the interval algebra.

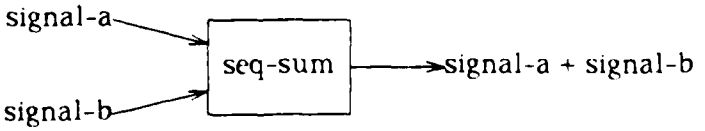
Signal Representation

Like SPL, KBSP represents signals as objects with functions for computing their samples and other properties. The basic properties supplied by all signals in KBSP are support and period.

Signal Definition

In KBSP one defines an entire class of signals at once. However, rather than think of such a thing as a definition of a class, one can think of it as the definition of a "system". In the case of Hamming for example, the task is one of defining the system *Hamming* which when applied to a scalar will generate a Hamming signal. For addition, one can define the system *seq-sum* which when applied to two signals, creates a new signal whose values are given by the sum of the two inputs.





This is not functionally different from the concept of a class with a finder as Kopec

defined it, but the notion that one is defining a system and that once defined one plugs the outputs of systems into other systems is a useful way to picture program development.

Reasons for Delayed Evaluation

Kopec justified delayed evaluation as follows

> "There is a clear conceptual distinction between the processes of defining a function (by constructing an expression for its values), and computing the function, by evaluating the expression at a specific point of its domain.
>
> This observation suggests that creating the program representation of a discrete-time signal or system should not imply the immediate creation of (representations for) all of its domain range pairs. ... creating a signal should not imply computing all of its samples."

There are other reasons for separating definition and computation beyond the conceptual distinction between them:

- If signals with infinite domains are to be represented, then delayed evaluation is essential.
- If there is some chance that the entire signal may not be needed, for example if it is to be decimated, then computing all the samples at definition time would be wasteful.
- If the definition of the signal might be changed to an equivalent but more efficient form before computation is necessary[58].

Transparent Caching

In the original description of SPL[57] it was pointed out that efficiency could be gained if a signal type called a store were defined, a signal that saved its samples after computing them. By inserting such stores, the programmer could avoid repeated computations if they fetched samples more than once, for example if they used the same Hamming window for two different purposes.

This idea was extended independently in both KBSP and later versions of SPL (such as SRL[59]) to the idea of transparent sample caching. With this capability, the system automatically saves any computed samples in case they are needed later. In SRL, any fetching of a signal caused the signal to be cached over its entire domain. With infinite domain signals this is not possible, so it is necessary in KBSP to allow caching of a subset of the domain. An additional advantage of caching only a subset is the efficiency that accrues from not computing samples that are not needed.

Discussion

The KBSP package makes algorithm development more convenient. This enhanced the development of the PDA by decreasing the "turn-around" time between an algorithmic idea and an implementation. It also served as the basis for symbolic distributed representations in the PDA. For example, the input transcript was implemented as an augmented sequence object whose sample values were symbolic. This allowed the graphical interface built into the KBSP package to be used to examine the state of the PDA.

The KBSP was also used by the Knowledge Manager to compute the confidence sequences for distributed symbolic assertions and the probability density sequences for f0 assertions. For those two applications, the KBSP proved invaluable.

For a more complete discussion of the principles and implementation of the KBSP package, the reader can refer to the PhD thesis of Cory Myers[58].

4.2. The Epoch System

The PDA has a special datatype ("epoch") to represent temporal events. The motivation for developing a separate datatype to represent temporal events was as fol-

lows:

- A single number representation for temporal locations failed to capture the inherent temporal uncertainty reflected in both symbolic input and numerical measurements.
- The information needed to refine the statistical information about event positions is gradually acquired during system operation. So it was apparent that extensive manipulation of event estimates would be necessary.
- Besides the numerical time of the event, there was a concept of neighbors to that event that needed representation. There had to be an object type to which the distributed assertions such as phonemes would be attached.
- Given the overall PDA philosophy that the user can interact (add and retract information) during system operation, this connection datatype had to be *"smart" enough to be able to undo its effects on demand.*

There are four basic tasks handled by this system of epochs:

- Compute position and uncertainty statistics for composite epochs that represent multiple event estimates.
- Determine when individual simple epochs should be merged with other simple or composite epochs.
- Support the rule system by efficiently answering questions concerning the relationships between temporally distributed assertions.
- Provide backtracking capability to allow previous merges to be undone.

Basic Epochs

The basic representation of an epoch is a data object with the following structure:

Name
: The name of the object is generated automatically by the system. For simple epochs, the name will be "se#" where # is a number. For composite epochs it will be "ce#". The name is a LISP symbol whose value is the epoch. This allows the operator to examine epochs individually for development and debugging purposes and for studying the systems conclusions about events.

Mean
: This is a number (in samples) that specifies most likely position of the "event" this epoch represents.

Variance
: This number reflects the temporal uncertainty in the Mean.

Neighbors

This is a set of links (pointers) to the temporal assertions that have a boundary at this epoch. For example, a phoneme-mark has an epoch as its "start" and an epoch as its "end".

Types of Epochs: Simple and Composite

When a new epoch is needed (e.g. a new assertion is to be made whose endpoints are not given by earlier assertions), a simple epoch is created. This happens as a result of rule actions making new distributed assertions (e.g. the numeric-voiced rule makes epochs to start and end voicing assertions).

When multiple simple epochs lay near one another, the Epoch system constructs a composite epoch to represent the group of simple ones. The composite shadows or hides the simple epochs it represents. Requests given to the simple epochs are forwarded to the composite for handling. This means that for most purposes distributed assertions whose simple epochs have been merged appear to share a single composite epoch. The merging of epochs and resulting effects are depicted diagrammatically in figure 4.1 The boxes represent distributed assertions and the circles represent epochs. The original configuration has two phoneme marks each with a simple starting and ending epoch. Assuming that *se* 24 and *se* 25 are sufficiently close to one another, they will appear to merge into a single epoch (*ce* 3) connecting both assertions (as in the middle part of the figure). In actuality, the simple epochs still exist, they are the simply forwarding all requests for information to *ce* 3 for reply (this is depicted in the bottom part).

Simple Epochs

Simple epochs are made by rule invocations when they need to refer to a place in time. For example, the rules which parse the input transcript assert phoneme-marks

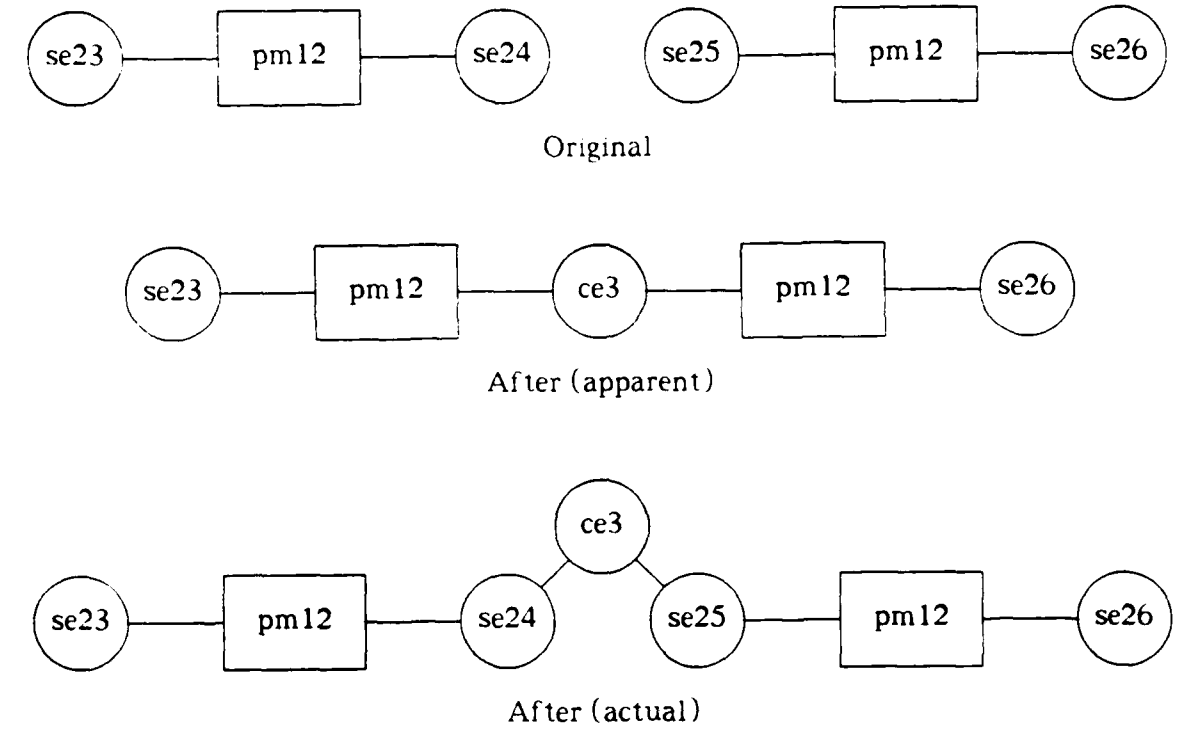


Figure 4.1 The Merging of Epochs

each of which starts and ends on a different simple epoch. The Epoch system subsequently merges the adjacent end/start simple epochs of adjacent phoneme-marks (because of their closeness) effectively connecting the phoneme-marks together.

Simple epochs are not created when a distributed assertion is built from epoch information that already exists. For example, when syllable-marks are built from their phoneme-marks new simple epochs are unnecessary (and inappropriate). Instead, the syllable-marks are made to start and end on the same simple epochs their included phoneme-marks start and end on. This is necessary because the computation of composite epoch statistics makes the assumption that the individual simple epochs contribute independent information about the underlying event position.

The neighbors of a simple epoch are all the distributed assertions that start or end on it. Typically, when first created a simple epoch has one neighbor. For example, initially the starting epoch of a typical phoneme-mark has only that phoneme-mark as a neighbor. However, if a syllable-mark is added that starts with that phoneme-mark, then that syllable-mark will become another neighbor of the epoch. This is the only way that simple epochs ever get neighbors. They *never* get them through merging. That is the purpose of composite epochs.

Simple epochs are unaffected by merging so retraction can be performed. By keeping a record of what contribution a given simple epoch makes to a composite (which neighbors it contributes and what event estimate), it is possible for the Epoch system to correct the state of the composite if the simple epoch is retracted.

Because simple epochs are the only type that are directly connected to assertions, they are the only type that can be retracted. Composite epochs are only attached to simple epochs. Each simple epoch depends on one assertion, typically the one it was created for. For example, the simple epochs on which a numeric-voiced assertion starts and ends depend on that assertion. If the assertion is retracted, the epochs are as well.

In cases where simple epochs can have several neighbors another object may provide the support. For example, the simple epochs that terminate the transcript marks (phoneme-marks, syllable-marks, etc.) depend not on individual marks, but on the transcript itself (despite the fact that the transcript object doesn't use them for anything). This way, it is possible to retract and replace a phoneme-mark or a syllable-mark without retracting the epoch and all the other marks it is attached to.

Once a simple epoch becomes part of a composite epoch, all requests to that simple epoch (e.g. "what is your mean", "who are your neighbors") are forwarded to the composite epoch. In this way the simple epoch "takes on" the properties of the composite it is a part of.

Composite Epochs

Composite epochs are made only by the Epoch system (in order to merge together simple epochs). They are not directly attached to assertions, so they only receive inquiries that are forwarded from simple epochs. Each merged group of simple epochs is represented by one and only one composite. Such simple epochs have a record of their composite and the composite has a record of its constituent simple epochs.

The composite does not depend on any individual simple epoch out of the group, but will only continue to exist as long as there are at least two constituents. If the group falls below this number, the remaining constituent reverts to answering requests itself and the composite is removed.

The neighbors of a composite are the union of the neighbors of the simple constituents. Because of this and the fact that simple epochs forward requests to the composite, when two simple epochs are merged the assertions to which they are attached will become neighbors. This is how much of the connectivity is established in the network of assertions.

Computing Statistics

The statistics of a composite (mean and variance) are established from the statistics of the constituents in a straightforward manner. By assuming that each simple epoch contributes a Gaussian independent noisy observation of an underlying event,

the constituent statistics are given by:

$$\bar{c} = \frac{\sum_i \frac{\bar{s}_i}{\sigma_i^2}}{\sum_i \frac{1}{\sigma_i^2}} \tag{4.1}$$

$$\sigma_c^2 = \frac{1}{\sum_i \frac{1}{\sigma_i^2}} \tag{4.2}$$

where $\{s_i\}$ is the set of estimates, $\{\bar{s}_i\}$ are the estimate means, and $\{\sigma_i\}$ the estimate variances.

Controlling Merging

Each time a newly created simple epoch is close to an existing simple or composite epoch, a decision must be made whether to merge the two. Both numerical and symbolic issues impact on this question. From both perspectives, the program must decide if the merger is appropriate.

From a symbolic standpoint, the only criterion the PDA is interested in is that the epochs to be merged have no neighbors in common. The only way they could would be if they were starting and ending epochs of a single distributed assertion. Thus, this criterion serves to prevent the starting epoch of an assertion from being merged with its ending epoch, an action that is clearly unreasonable since the two epochs necessarily represent different events.

From a numerical standpoint, the PDA is interested in how probable it is that the proposed epochs refer to the same event. The need to answer this question is the reason that all epochs must specify statistics. Without some statement about where the underlying event might be (some statement about the uncertainty of the epoch), it

would not be possible to answer that question. If the statistics were assumed a priori to be the same for all simple epochs, then the system would be unresponsive to the available information about epochs; information available from the operator (about transcript epoch statistics), from symbolic modules (about the likely timing of stops) and from numerical modules (about the uncertainty in power based voicing assertions).

Given the availability of simple epoch statistics and the assumption that each epoch proposed for a merge is an independent Gaussian observation of *some* event, the formula that gives the probability density for that configuration of simple epochs stemming from a single event is

$$p(\vec{s}) = \frac{1}{\pi^{\frac{n}{2}} |\Lambda|^{\frac{1}{2}}} e^{-(\vec{s}-\underline{\vec{s}})^t \Lambda^{-1} (\vec{s}-\underline{\vec{s}})} \tag{4.3}$$

where $\vec{s}$ is the vector of simple epoch means, $\underline{\vec{s}}$ is a vector whose elements are all equal to the mean value of the proposed composite, and Λ is a diagonal matrix of simple epoch variances.

To rate the quality of a proposed cluster of simple epochs in absolute terms, the PDA integrates this probability density over all larger clusters, where the size of the cluster is given by the "Mahanalobis distance"[60] (normalized radius) of the cluster R.

$$R = \left| \frac{\sum_{0}^{n-1} \frac{(s_i - \underline{s})^2}{\sigma_i^2}}{n-1} \right|^{\frac{1}{2}} \tag{4.4}$$

Numerically then, the PDA's criterion for merging a cluster of simple epochs is the likelihood that another randomly chosen cluster would be larger in "radius". The

PDA uses an absolute numerical threshold of 20% to make this decision. That is, if there is at least a 20% chance that another randomly chosen cluster (with the same number of elements) would have a larger radius than the proposed cluster, then the proposed cluster is merged. Otherwise, the proposed epochs are considered estimates of different underlying events. This value was chosen empirically to balance the number of incorrectly clustered epochs with the number of failures to properly cluster.

Supporting Rule Conditions

By linking assertions to one another, the Epoch system may satisfy certain rule conditions. For example, if a rule is interested in an /s/t/ phoneme cluster, then when the Epoch system merges the ending epoch of an /s/ phoneme-mark with the starting epoch of a /t/ phoneme-mark, that rule should be activated.

This is accomplished in the PDA through change propagation. All neighbors of a simple epoch are recorded as "users" of the epoch. If the epoch's state changes, the neighbors (assertions) must be "informed". The Knowledge Manager is in turn a user of every assertion. When a simple epoch is merged, it is considered changed. It informs its users (i.e. its neighbors) and they inform the Knowledge Manager. The Knowledge Manager has a record of what object types each rule is interested in, and it informs appropriate rules that the assertions connected to that epoch have changed. Finally, these rules reevaluate their conditions with respect to the changed objects and determine if firing is appropriate.

Besides stimulating the reevaluation of rule conditions, the system of epochs provides an efficient mechanism for evaluating rule conditions concerned with adjacency. Because of the direct linkage between assertions and simple epochs and the linkage between merged simple epochs and composite ones, a rule can evaluate a condition

concerning adjacency by "chasing pointers" rather than with an implied search of the assertion database.

For example, in the previous case of the /s/t/ context, in the absence of this linkage the rule might be written as shown in figure 4.2. With the linkage created by the epoch system the condition can instead be written as shown in figure 4.3, where the (*right –neighbor* x) form looks up y using the epoch linkage. If there are n /s/s and m /t/s in the utterance, the comparison clause (*eq* (*end* x) (*start* y)) in the rule of figure 4.2 would be run $n \times m$ times. None of the clauses in the rule of figure 4.3 would be run more than n times.

```
(defrule <slow>
  CONDITIONS
  (type 'phoneme mark x)
  (isa ':s x)
  (type 'phoneme mark y)
  (isa ':t y)
  (eq (end x) (start y))
  ...)
```

Figure 4.2 Inefficient Condition

```
(defrule <fast>
  CONDITIONS
  (type 'phoneme mark x)
  (isa ':s x)
  (let y (right neighbor x))
  (isa ':t y)
  ...)
```

Figure 4.3 Efficient Condition

Backtracking

Backtracking is the process by which an assertion is retracted from the system and its effects undone. The design of the Epoch system is such that the merging of simple epochs can be undone if one or more of the simple epochs is retracted. To accomplish this, the Epoch system preserves each constituent (simple) epoch untouched when a merge is performed.

The only effect of the merge is to cause each simple epoch involved to forward requests for information to the composite epoch representing it. In that way, the simple epoch appears to have the properties of the composite (mean, variance and neighbors) while the original properties of the simple epoch are preserved. Because all the simple epoch information is preserved, it is possible to retract any given constituent epoch and recompute the state of the composite properly.

A Demonstration

Figure 4.4 shows the operation of the Epoch system by comparing the same set of assertions with and without merging. The upper part of the figure shows the realigned transcript and the voicing marks with epoch merging disabled. The lower part of the figure shows what happens if merging is allowed. Vertical lines indicate connected epochs.

In the unmerged case, all information that is phonetically derived shares common epochs. Deriving one phonetic description from another (e.g. voicing marks from phonemes) does not generate new boundary information except in the case of the frication/aspiration boundary during stop release. However, there are no connections between numerically derived marks, nor between phonetically and numerically derived ones. Therefore, there has been no refinement of boundary position estimates.

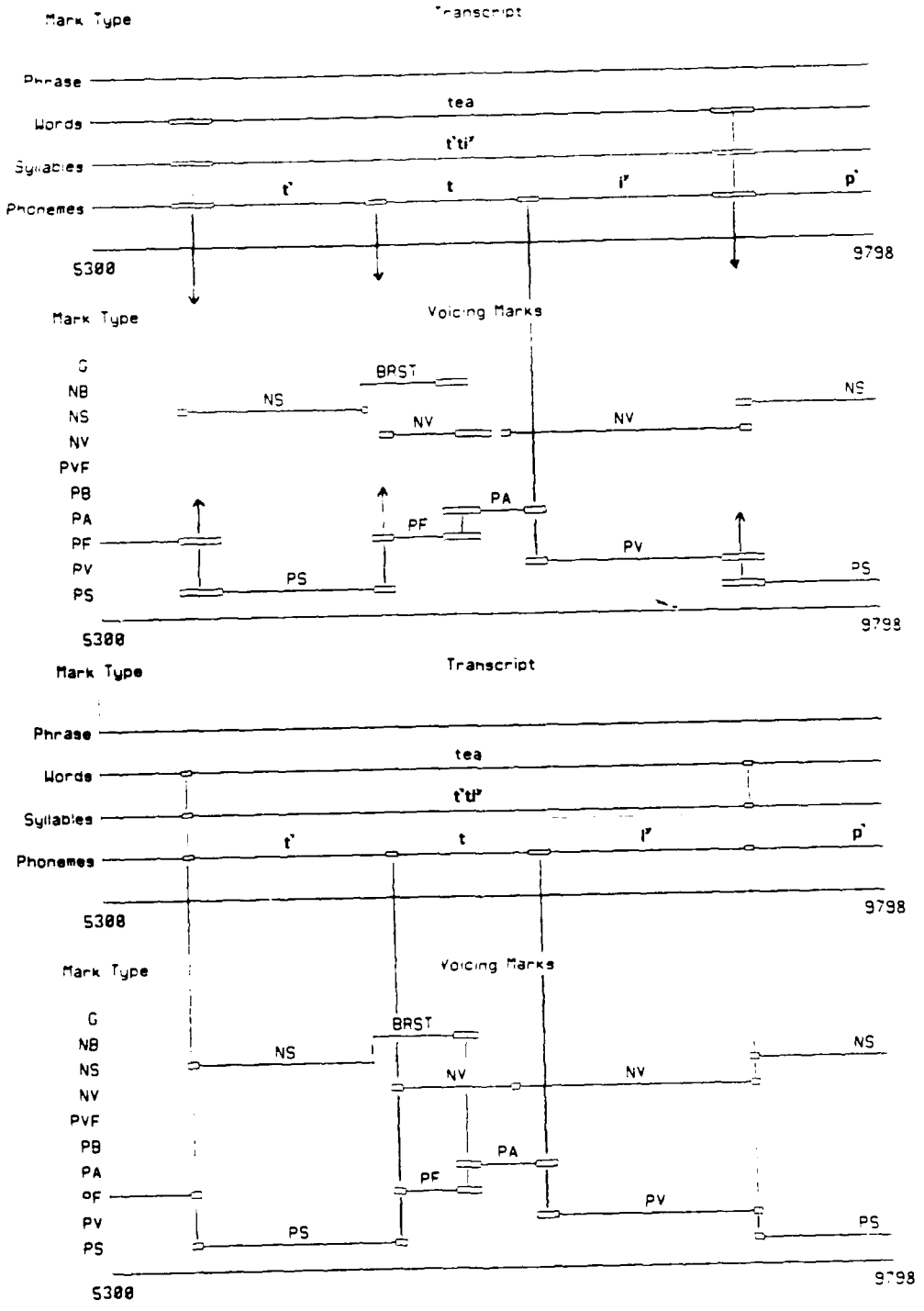


Figure 4.4 The Effects of Merging

In the merged case, there is extensive linkage between the marks and those which have been merged show substantially reduced deviation; a reduction in uncertainty that results from combining multiple sources of information.

4.3. The Rule System

4.3.1. Overview

The rule system in the PDA consists of a set of rules and a set of procedures that implement rule behavior. The basic philosophy of the PDA rule system is as follows:

- A given rule is "primed" to respond to some configuration of data in the PDA. If that configuration arises then the actions part of the rule is executed, once for each such configuration.
- If at a later time, the data configuration which triggered the rule ceases to exist, then the changes caused by that rule firing will likewise be undone.
- Similarly, if a rule is retracted then the changes caused by that rule's actions will be removed.
- Finally, anytime a new rule is entered into the system it is informed of the current data in the PDA and executed on all appropriate data configurations.

The goal of this philosophy is a rule system whose rest state[1] is dependent on the rules and data present, but not dependent on the history of rule and data entry. Ephemeral rules and data should not effect the rest state of the system, nor should the order of entry of rules/data actually present.

This approach can be contrasted with rule systems such as YAPS[61] and OPS-5[62] where the rest state is dependent on history. In these systems:

- The disappearance of data configurations causes no automatic retraction of rule results, so a transient data configuration can cause a difference in the rest state.

[1] The rest state is the state of the rule system that is achieved once all rule firing and consequent state modification have ceased.

- The retraction of rules does not undo their effects, so a transient rule can also affect the rest state.
- Any data entered prior to a rule is invisible to that rule, so the order of rule and data entry can influence the rest state.

A History Independent Rule System

The advantage of a rule system whose results are independent of the history of data entry is simplicity of behavior. It is easier to understand the PDA's behavior if that behavior is determined by the information that is present, not the order in which that information was entered, nor any information that is no longer in the system. Similarly, the advantage of a rule system whose results are independent of the history of rule definition and data entry is that this simplicity of behavior is preserved even under changes to the rulebase.

One motive for developing a history independent rule system is convenient run time interaction, appropriate for an assistant. Such interaction can take the following forms:

- To understand the contribution of one particular type of information, the operator may enter information in batches. For example, the operator might input phoneme-marks, observe the results, then enter syllable-marks to see the impact of this additional information.
- The operator might alter the transcript because they want to experiment with the effects of different hypotheses. For example, the operator might change a phoneme-mark from one type to another.
- To determine the effects of a rule change, the operator can run the PDA up to a rest state, make the change and run the system to rest again, observing the results. Proposed corrections to erroneous rules can be evaluated in a like manner.

It is easier for the operator/rule developer to understand, manipulate and develop the PDA if the conclusions of the system are dependent on the current configuration of rules and data, and not on how that configuration was arrived at.

Another advantage of a history independent rule system is in the impact of rules which change the input data. One rule was written that verified stop voice onset time (VOT) against a table. If a word-final stop-mark was found to have excessive VOT, a word pause was inserted after the stop-mark in the input transcript. Since this rule might be run at any time after the stop had been entered, there was no way for the operator to know if results might be derived from the initial stop configuration which would be invalid after the change. However, because the PDA rule system automatically retracts any such results, the operator need not be concerned with them.

The computational price one pays for such history independence is the cost of maintaining a network of dependency links between objects, and propagating effects along the network. The cost in terms of programming and debugging such a network is also high.

No Control of Rule Order

Another feature of the PDA's rule system is a lack of conflict resolution. Most rule systems have a set of pending rules whose conditions have been satisfied, but which have not yet been fired. These rules are considered in "conflict". Typically, some "conflict resolution" strategy is used to select a single rule from this set, that rule is executed, and then conditions of all rules are reevaluated. The conflict resolution strategy impacts the behavior of the system, because the firing of one rule may prevent other rules from firing. Thus, programming such systems involves understanding and possibly controlling this conflict resolution strategy.

The rule system in the PDA has such a set of pending rules, but instead of selecting one, all pending rules are fired. This approach was chosen because we felt that the use of a conflict resolution strategy as a way to influence system behavior was

undesirable because of its subtlety. For example, the idea that the order in which rules appear in the rulebase should affect system results seemed undesirable. Thus, the firing of rules in the PDA must be controlled explicitly with the conditions of the rules and not indirectly through conflict resolution. Also, since the rule system behaves in a history independent fashion, the order of rule fire is largely irrelevent.

4.3.2. Architecture

A picture of the relationship between rules and the Knowledge Manager (KM) is shown in figure 4.5. The KM is connected to all rules and knows which object types each rule is interested in. New assertions (generally made by rules) are handed to the Knowledge Manager (KM) for processing. The KM passes those objects on to the appropriate rules so the rules can check their conditions. In addition, the KM makes itself a "user" of each new assertion, so it will be informed about any change in the

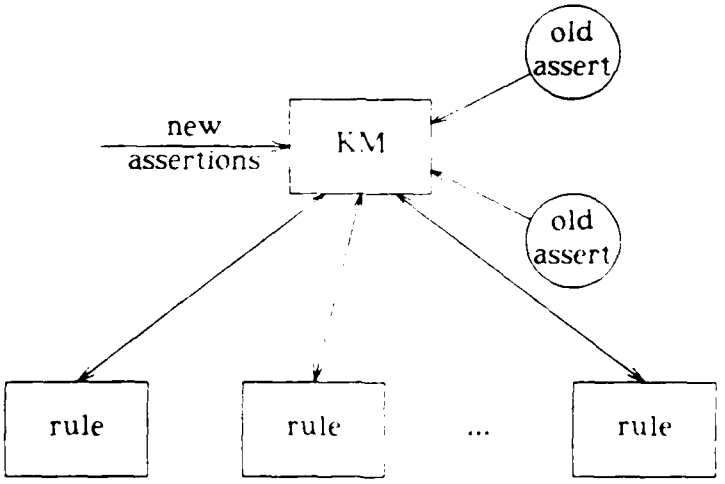


Figure 4.5 Rules and the Knowledge Manager

status of the assertion. Such changes cause the KM to inform appropriate rules so the rules can verify their condition status with respect to the changed assertion.

A more detailed account of how the KM handles this task and how rule actions are executed is given below in the section concerning the KM. The rest of this section is concerned with the procedures and structures that accomplish rule condition evaluation.

Rule Conditions

Rule conditions are composed of three types of clauses:

Match Clause
: The clause (*type* 'phoneme-mark x) is a match clause. *X* must be an asserted object of type 'phoneme-mark.

Let Clause
: The clause (*let* y (*right –neighbor* x)) is a let clause. When the actions are performed, y will be bound to the value of the expression (*right –neighbor* x) for the choice of *x* given above.

Test Clause
: The clause (*isa* ':stop x) is a test clause. *X* must be stop phoneme for the rule to be triggered.

The match clauses specify both that a given variable should bound to new assertions, and that only assertions of a given type need be considered. Thus match variables do for this rule system what pattern variables do in a pattern based rule system. They implicitly iterate over new assertions. However, in this system one must explicitly declare these variables rather than simply mentioning them in a pattern, and in this system such variables have a type which limits the things they may be bound to. Whenever a new object is asserted, the KM passes that object to all rules containing type clauses for an object of that type.

Let clauses are another way of binding variables, in this case, the variable's value is uniquely determined from the values of variables mentioned in the body of the clause. Unlike match variables, these variables do not independently iterate over new assertions. Let clauses serve to simplify the expression of rules by allowing a variable to be assigned to an often used value. Also, they can be used to save the value of some expression for use either in the PREMISE-ODDS portion of the rule or in the actions. Finally, because the binding that is created when an rule is triggered is assumed to be dependent on the value of all variables in the conditions, assigning a variable to the value of some expression is a way of signifying such dependency. Variables mentioned in let clauses are called "local variables". Since match clauses and let clauses both assign variable values, only one of the two methods may be used with any given variable.

The test clauses are the primary way of constraining rule execution. Tests may involve arbitrary LISP functions of any local or match variables. The test clauses are affixed to a network that is used to determine if new (and changed) assertions should trigger rule execution.

Each rule is a test network

Each rule is responsible for two tasks. Given a new assertion, a rule must determine if it can derive new bindings from it. Given a changed assertion the rule must verify whether or not old bindings are still valid, and it must derive new bindings that are now possible as a result of the change. One way to understand this process is to picture condition evaluation as a multi-dimensional constraining process.

Each variable is an axis in a multi-dimensional space. Each possible choice for a variable is a value somewhere on that axis. Suppose a rule has match variables x and

y, each of which must be a phoneme-mark, and 20 phoneme-marks have been asserted. The space involved is two dimensional ([*x* ,*y*]) and each axis has twenty possible values, so there are 400 choices for the space [*x* ,*y*], all of which would cause the rule to be executed.

Suppose further that rule's conditions include the test (*isa* ':stop x) and only three phoneme-marks are stops (a /p/ and two /g/s). Then the number of choices for x drops to three and the total number of satisfactory choices for [*x* ,*y*] drops to 60.

Finally, suppose the rule includes the test (*eq* (*phoneme* y) (*phoneme* x)). Now there are only two choices for [*x* ,*y*] and the rule will only be fired twice. Constraints on the satisfactory choices for the top-level space of the rule (the one containing all variables mentioned in the conditions) are accomplished through the type restriction in the match clause, and through the test clauses.

Each rule uses a network structure of tests and stores to determine which combinations of match variable bindings are satisfactory. A typical network is depicted in figure 4.6. The boxes depict stores where choices for each subspace are kept (the boxes are labeled with the space they store). Each new assertion is added at the bottom of the network as a choice for a one-dimensional subspace, and the acceptable choices propagate upwards through the network.

Each new choice must pass through tests (depicted by diamonds). If a choice fails a test at some level, the choice is diverted into a store for bad choices. If the test is passed, the choice is saved in the store for good choices (above the test) and continues to propagate up the network.

Dots where paths join are called "merges". Each time a new choice reaches a merge, it is joined once with every compatible choice present in the good-store on the

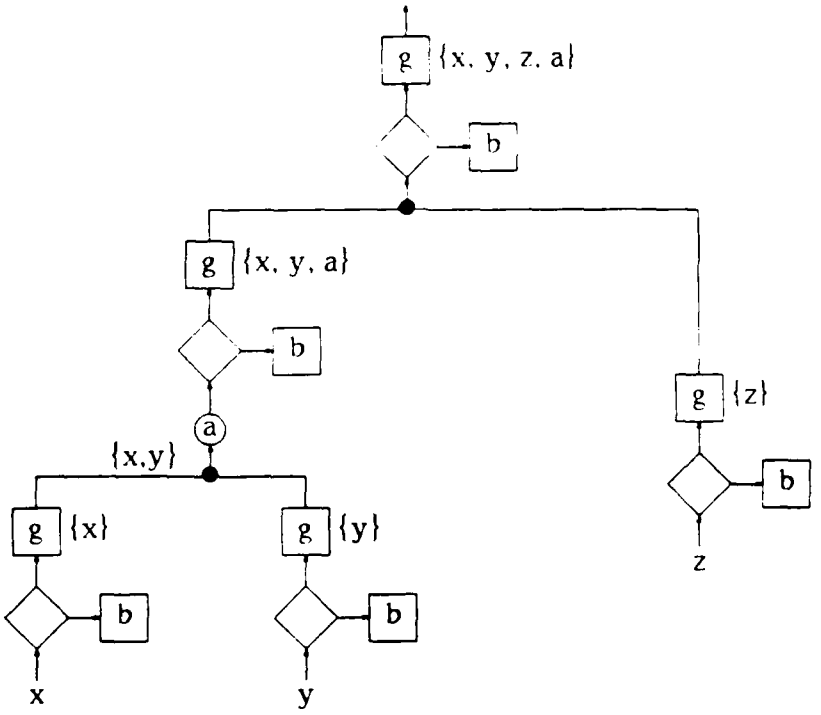


Figure 4.6 Typical Rule Condition Network

other entering branch. This typically generates many choices for the merged subspace, which continues to propagate up the network. Choices from two branches are compatible if the projection of each choice onto the common subspace has the same value.

The circles represent the let clauses. Each of these adds a new dimension to entering subspace (at the bottom of the circle) and produces the value along that new dimension by invoking the let body.

Efficiently Evaluating Conditions

The overall goals of the rule system are the correct and efficient evaluation of rule conditions. One way to achieve correct evaluation would be to find all the new top-level choices that result from each new assertion, and test them. This would

correspond to moving all test diamonds up to the top of the network. However, such an approach would be extremely inefficient. If a test can be performed on a subspace before a merge, then doing it there once is cheaper than doing it after the merge on the many choices that result.

By pushing tests as far down in the network (to the nodes with the fewest dimensions at which the test can be performed) one gets the most power from them. Take any one test and consider how many final choices are eliminated by each invocation. If the rule's space is x, y, w, z, but the test only requires x and y, then each time the test is failed by a choice x_0, y_0 in subspace x, y, all the final choices x_0, y_0, w, z will have been eliminated.

Other rule systems build networks similar to this to evaluate rule conditions. For example, YAPS builds structures using a similar branching architecture as shown in figure 4.7. In YAPS, the network topology is completely specified by the condition patterns, and is influenced by pattern order. Each test in YAPS (tests are specified separately from patterns) is affixed to this network in the first location where all variables in the test are available, scanning the network from left to right, bottom to top.

While this approach places tests low in the network, it has a flaw. Because the topology of the network is specified by the rule patterns and not the test subspaces, it may be the case that a test's subspace is not present in the network. That will mean the test gets performed on a higher dimensional subspace which will likely lead to unnecessary testing. An example of this problem is shown in figure 4.8. This rule finds all sequences of three NUMs in a row. The numbers adjacent to the branches in the figure are the number of choices that will have flowed through that branch assuming the facts (NUMBER 1) through (NUMBER 100) were asserted. One million invo-

```
(yaps rule (x  x) (y  x) (z  z)
        tests (test 1  x  y) (test 2  y  z)
        actions ...)
```

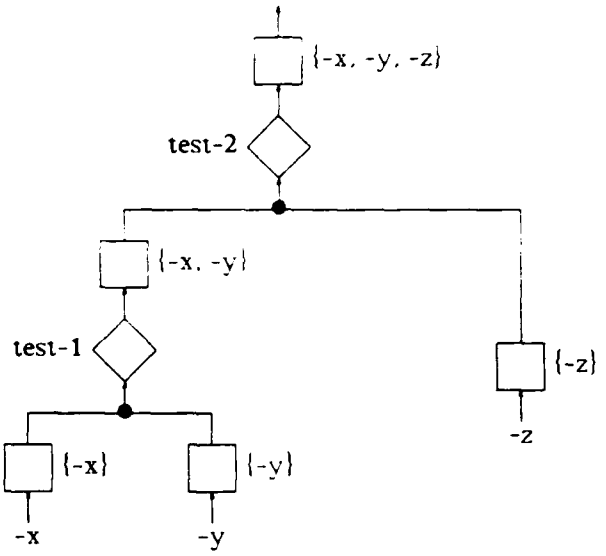


Figure 4.7 YAPS Network Architecture

```
(yaps-rule (number x) (number x) (number z)
        tests (eq z (+ 2 x)) (eq z (+ 1 y))
        actions ...)
```

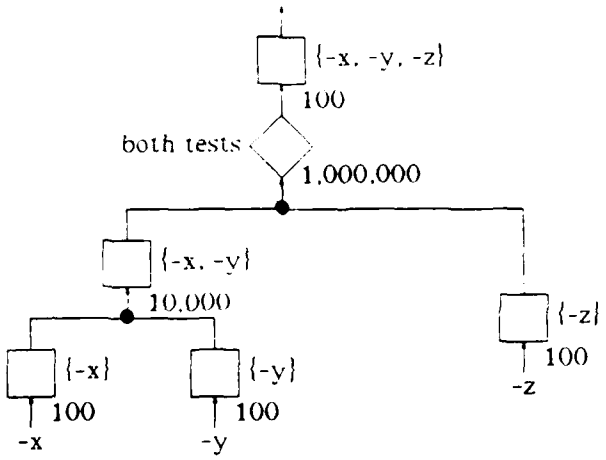


Figure 4.8 Inefficient YAPS Network

cations of the test will occur even though only ten thousand would have been necessary.

The PDA rule system differs from YAPS in its approach toward generating the network. Rather than derive the network directly from pattern clauses without regard to the subspaces they represent, the PDA designs the rule network to contain all test subspaces so all tests can be used with maximum efficiency.

Another source of efficiency is in the "merging" process. If the store for each subspace is indexed by the shared subspaces of all stores it is combined with, then when a new choice comes in from one of those stores, the set of compatible choices in this store can be found with a single lookup (by projecting the new choice onto the shared subspace and indexing). Without this index, it is necessary to do a linear search whose cost will be proportional to the contents of the store.

Handling Change

Because the state of objects can change, it becomes necessary to reevaluate tests. When a rule is told about a changed object, first it scans up the network stores looking for choices with local variable values that were determined by the changed object. These choices may be eliminated if the local variable value is different as a result of the change. When that happens, all choices that depended on that choice are also eliminated, and a new choice with the correct local variable value is installed in the bad-store for that subspace.

Now the good-stores with associated tests are checked for choices that use the changed object. Those choices are verified and any that fail are flushed (along with all dependents). Finally, the bad-stores are checked for choices that use the changed object (including choices just generated through a local value change). Those that now pass are propagated up the network as if they were good choices.

4.4. The Knowledge Manager

The Knowledge Manager (KM) is a name for the parts of the PDA that handle the interactions between assertions and manage the operation of the rule system. The tasks of the KM fall into three categories:

- Run the Rule System
- Operate Dependency and Support Networks
- Compute Confidence and Other Statistics of Assertions

4.4.1. Running the Rule System

The preceding section on the PDA rule system focused on the determination of condition satisfaction. Rules really have two independent roles to play in the operation of the PDA. One role is played by the rule's conditions (which are represented for

each rule with the network described earlier). As this network is fed information about new and changed assertions it is able to generate new (or verify old) binding objects. The second role played by each rule is that of generating new assertions and modifying old ones, the effects of the ACTIONS. In the PDA, these two roles fit into separate phases of the overall system cycle as follows:

1) Execute rule actions using bindings which have yet to be used to fire a rule. Keep track of object change and old bindings affected by it. Collect newly generated bindings for execution on the next cycle. Mark bindings dependent on changed objects as potentially invalid.
2) Check rule conditions against changed objects, determine whether bindings affected by change are still valid.
3) Kill bindings that are not verified. Retract all their results (possibly including other bindings both executed and unexecuted).
4) Check rule conditions against new objects.

This cycle is one job of the KM. It is run until bindings stop being generated (the so called "rest state" of the system). The implementation of this aspect of the KM is a matter of keeping tables of unfired bindings and unverified bindings, newly generated or changed assertions, and rules currently in the system.

4.4.2. Network Management

There are two networks managed by the KM. One is a network of dependency, of objects that rely on the existence and status of other objects. The other is a network of support, consisting of links that signify added or reduced belief in an object or information about its statistics.

While these networks are separate from the KM proper, the KM uses them to update the status of the PDA when changes in the status and confidence of assertions have taken place.

Dependency Network

The dependency network is a way of maintaining the PDA in a self consistent state despite the modification and retraction of data and rules. It was developed to support the idea of a history independent rule system mentioned above. When there is an implicit dependency between one object and another (for example between a transcribed phoneme and the assertions it led to) then the dependency network can be used to express that dependency. One object is said to be the "user" and the other the "server".

If the server is retracted then all its users are notified and they are (usually) retracted as well. Thus the user/server relationship expresses one object's dependency on another. For example, a rule binding is a user of all the variable values for that binding. These values might include the waveform, some phoneme-marks or syllable-marks, the VOICED assertion, etc. If any of those objects are retracted, then that binding is retracted as well.

This type of processing is modelled after the processing of so called truth maintenance systems (TMS)[55]. Those systems maintain a network of nodes each of which can be in one of three states (on, off, or unknown). The network connects those nodes with constraints that correspond to logical "or" and logical "not". In operation, one specifies the value at a subset of the nodes, and the network drives the remaining nodes to values as necessary to satisfy the constraints (or the network reports a contradiction signifying that the set of specified values is inconsistant with the network topology).

The propagation of retraction information by the Knowledge Manager is exactly the sort of processing that takes place in the operation of a TMS. What we have done

is to merge that technology with the technology of rule systems to create a rule system that is history independent and consequently easier to program. In doing this we had to extend the nature of the propagated information.

In a TMS, each node has only three states. In the PDA (as is likely in any KBSP problem) the nodes have more complicated states because they represent more than simple logical quantities. The assertions in the PDA represent vocal excitation modes, f0 estimates, etc. Therefore, besides the logical dependency that is manifested by chains of retraction, objects can depend on one another in ways that don't involve being true or false, valid or invalid. For instance, a word mark depends on the syllable marks it covers. If the epoch that ends the last syllable of the word is changed, then the epoch that ends the word mark must also be changed.

To handle this type of dependency we developed the idea of propagating change information. When a server is changed, it broadcasts a message indicating that it has changed to all of its users. The users can then react to the change. This aspect of the user/server relationship allows the PDA to react to modifications that fall short of retraction.

Any binding is notified if objects it is using are changed. The binding's response to this change is to put itself on a list of possibly invalid bindings. The next time phase 2 of rule system operation occurs ("check changed objects"), the KM will ask this binding's rule to test its conditions against the changed object. If that test passes, then the rule notifies the binding, and the binding removes itself from the invalid bindings list. If the test fails, the KM retracts the binding and the effects of that retraction ripple through the dependency network retracting all objects that were dependent on the binding (users, users of users, ...).

The binding responds to both change and retraction on the part of its servers. The KM responds to change but not retraction. The KM makes itself a user of all new assertions, so it can be aware of those which change and should be checked against rule conditions. However, when an assertion dies the KM merely forgets it. The opposite behavior is exhibited by the rule condition networks. They are users of all the objects that are held in the stores of the network. Changes in the objects are ignored. However, retraction of the objects causes them to be flushed from the stores.

The structure of the dependency network is implemented by pointers from users to servers and vice-versa. Each server keeps a list of users and each user keeps a list of servers. If the user is retracted, it notifies the server and the server disconnects from the user. If the server changes, it notifies the user so the user can respond to the change. If the server is retracted it notifies the user which almost invariably is retracted also.

Support Network

This network represents and updates lines of support between assertions. The two ends of a support link are the "supporter" and "supportee". The supporter is providing information about the value of the supportee. In the case of symbolic assertions, this support affects the confidence in the supportee. For example, the unique assertion VOICED is supported by all other voicing related assertions (phonetic-frication, numeric-silence, ...). These supporters are what determine the confidence in VOICED as a function of time. In the case of numerical valued assertions, the supporters are providing information about the value. The FINAL-PITCH assertion is supported by pitch assertions derived numerically and symbolically. Each supporter supplies an estimate of f0 together with confidence and variance that contributes to

the probability density for f0 contained in FINAL-PITCH.

Figure 4.9 shows the typical support relationship for a symbolic assertion like VOICED. Several sources of support must be combined together (using procedures described below) possibly together with a default support (explained below). The network preserves the connections between objects so if a new assertion is added that affects the confidence of some object, that change in confidence can be propagated to supported objects and the rest of the support network brought "up to date" with the new information.

The default support is a special kind that exists for handling support that doesn't come from an object. The best example for this is the support given to numeric-voiced assertions. The procedure that creates them determines a value for their confidence in the process of analyzing the utterance. Unfortunately, this procedure is not an assertion itself, so it cannot participate in the support network. To deal with this problem, an assertion can have a default support which is static (unaffected by future change) that is supplied when the assertion is created. This is the way in which the numerical procedure provides support to the numeric-voiced assertions.

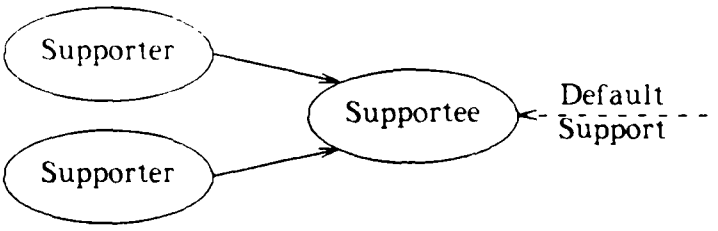


Figure 4.9 Typical Support Relationship

There are two ways to compute confidence for a symbolic assertion. One is used by bindings (the confidence in a binding stands for the confidence that the rule conditions were satisfied). They use the PREMISE-ODDS form in the rule to compute their confidence. This expression is given by the rule writer and may mention any variables in the conditions. The PDA system recognizes that all condition variable values that are mentioned in the PREMISE-ODDS form are supporters. Those that are not mentioned in the PREMISE-ODDS are not considered supporters (they do not influence binding confidence) though they are servers in the user/server sense. This is one reason for having separate networks.

The other way used by the PDA to determine the confidence in a symbolic assertion from its supporters is through a Bayesian probabilistic method similar to that used in PROSPECTOR and is described in detail below. This is the confidence computation method used by assertions. Since bindings support the assertions they create and since assertions are used as the variable values in rule conditions that generate and support bindings, the resulting support network appears as in figure 4.10.

The last type of support is for numerical assertions (f0). In this case, the supporters are pitch assertions that support the FINAL-PITCH assertion. Bindings were not used to convey support to FINAL-PITCH because the complexity of the support involved in numerical assertions is not handled properly by the BINDING object class.

In the PDA, these pitch assertions are not themselves supported. It is unfortunate and probably inaccurate that the confidence in the precursors to these pitch assertions (the waveform and the phonetic transcript) do not influence the confidence in the pitch assertions nor their support for FINAL-PITCH. That simply reflects the limited understanding we have at this point for how to deal with this problem.

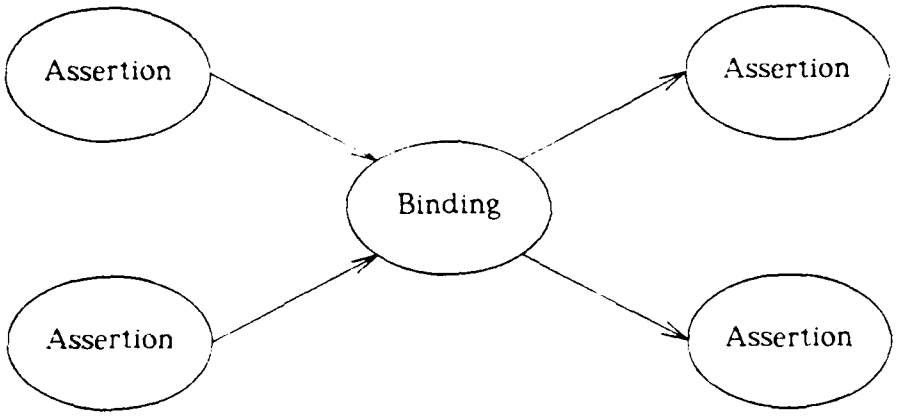


Figure 4.10 Alternation of Assertions and Bindings

The final point about support representation is that it is time-varying. Most assertions in the network are distributed over a substantial time interval. It was logical that their confidence be variable over that interval since the support for them would be variable. Therefore, the confidence of a distributed assertions is represented by a sequence object and computed with the help of the KBSP signal processing package. This combination of making a small number of assertions, but allowing them to vary confidence over time, was an effective solution to the problem of getting good temporal resolution of assertions such as VOICED, without resorting to making assertions every few samples (which would have been impossible with the rule system technology available to us).

4.4.3. Computing Confidence

The computation of confidence for symbolic assertions and probability density for numerical assertions is the last task of the KM. For bindings this is a straightforward task. The PREMISE-ODDS of the rule specifies the function that determines binding

confidence. For symbolic assertions the procedure is similar to PROSPECTOR's procedure for combining support, though the PDA performs this computation separately for each index in the domain of the assertion whereas in PROSPECTOR the confidence of an assertion is a scalar quantity. The procedure for combining support for numerical assertions is derived using probability theory with some independence assumptions. Because the mathematical form of the resulting densities is not a simple expression (such as a Gaussian), the resulting probability density for the numerical parameter is approximated with a uniformly sampled sequence over the region of non-zero probability.

PROSPECTOR

The derivation of the support combination rule in PROSPECTOR is as follows. Suppose there is an assertion receiving support r and assertions sending support s_i. First, assume that all that can ever be known about r comes from $\{s_i\}$, so

$$P(r \mid \forall) = P(r \mid \forall_{s_1}, \cdots, \forall_{s_n}) \ . \tag{4.5}$$

Where $\forall$ signifies "all that is known" and $\forall_x$ signifies "all that is known about" x. Further, assume that the probability of the supporters is independent of any knowledge about r, so

$$P(s_1 \cdots s_1 \mid r) = P(s_1 \mid r) \cdots P(s_n \mid r) \tag{4.6a}$$

if r is known to be true with certainty and

$$P(s_1 \cdots s_1 \mid \bar{r}) = P(s_1 \mid \bar{r}) \cdots P(s_n \mid \bar{r}) \tag{4.6b}$$

if r is known to be false with certainty.

For a single supporter s this leads to the following expression for the posterior probability for r.

$$P(r \mid \forall) = P(r \mid \forall_s) = P(r \mid s)P(s \mid \forall) + P(r \mid \bar{s})P(\bar{s} \mid \forall) \tag{4.7a}$$

$$= k_1 P(s \mid \forall) + k_2 P(\bar{s} \mid \forall) \tag{4.7b}$$

A typical graph of this relation is shown in figure 4.11. In PROSPECTOR, the rule writer specifies $P(s \mid r)$, $P(s \mid \bar{r})$, and $P(r)$ and the following formulas are used to compute k_1 and k_2.

$$k_1 = \frac{P(s \mid r)P(r)}{(P(s \mid r) - P(s \mid \bar{r}))P(r) + P(s \mid \bar{r})} \tag{4.8a}$$

and

$$k_2 = \frac{P(s \mid r)P(r)}{(P(s \mid \bar{r}) - P(s \mid r))P(r) + 1 - P(s \mid \bar{r})} \ . \tag{4.8b}$$

These expressions completely specify the relationship between $P(r \mid \forall_s)$ and $P(s \mid \forall)$, including the value of $P(s)$ the probability for s when nothing is yet known.

Unfortunately, s itself is often a supported node, so the rule writer will have provided a value for $P(s)$ even though there already was an implied value for it. The rule won't typically provide consistent information, so there is an inherent contradiction facing PROSPECTOR in deciding which information to use. To deal with this, the authors of PROSPECTOR chose to loosen the specification in (4.7a) in a way that is

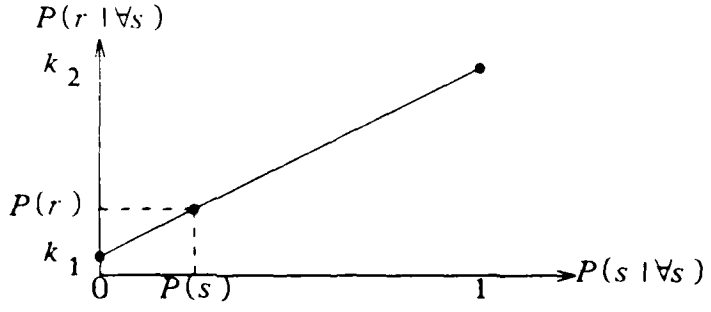


Figure 4.11 Support for r from s

depicted in figure 4.12. It assumes the values of $P(r)$ and $P(s)$ are correct and should be consistent. Likewise it assumes the values for k_1 and k_2 are computed correctly, and uses linear interpolation as shown to deal with the inconsistency.

To compute confidence from multiple supporters, PROSPECTOR uses the independence assumptions mentioned above and computes the odds of the receiver of support (where odds is defined by $O(r) = \frac{P(r)}{1-P(r)}$). The odds of the recipient is given by

$$O(r \mid \forall) = O(r) \prod_i \lambda_i$$

where

$$\lambda_i = \frac{O(r \mid \forall_{s_i})}{O(r)}$$

is computed using the single support expressions (4.7a) through (4.8b), together with the non-linear mapping implied by figure 4.12.

PDA

There are two ways in which the PDA differs from PROSPECTOR's approach to computing confidence for supported symbolic assertions.

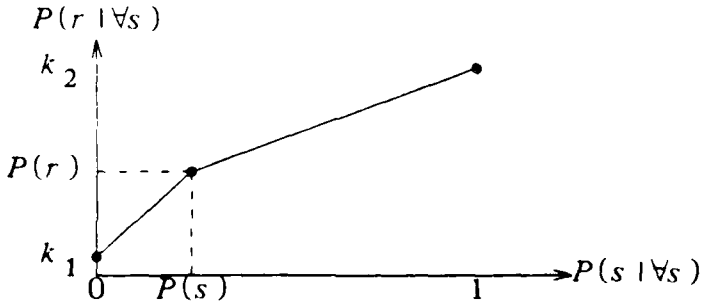


Figure 4.12 Modified Posterior Probability

1) The PDA computes confidence separately at each index.
2) The PDA uses a different representation for confidence that allows a much simpler (more economical) updating procedure so 1 is practical even though .5 million updates are not uncommon.

The source of the contradiction in PROSPECTOR is in asking the user for too many constraints. The PDA solution for this involves a different expression of the confidence of an object. Where PROSPECTOR uses the odds $O(r \mid \forall)$ as a measure of confidence for each assertion, the PDA uses the odds-factor $O(r \mid \forall)-O(r)$, the ratio of the current odds given what is known to the a priori odds. The formula for the current odds-factor of the recipient based on what is known about a single supporter is

$$\frac{O(r \mid \forall_{s_i})}{O(r)} = \frac{P(s_i \mid r)\dfrac{O(s_i \mid \forall)}{O(s_i)}+1-P(s_i \mid r)}{P(s_i \mid \bar{r})\dfrac{O(s_i \mid \forall)}{O(s_i)}+1-P(s_i \mid \bar{r})} = \lambda_i \quad , \tag{4.9}$$

or more simply

$$OF(r \mid \forall_{s_i}) = \frac{P(s_i \mid r)OF(s_i \mid \forall)+1-P(s_i \mid r)}{P(s_i \mid \bar{r})OF(s_i \mid \forall)+1-P(s_i \mid \bar{r})} = \lambda_i \quad , \tag{4.10}$$

and the formula for the odds-factor based on all information is

$$\frac{O(r \mid \forall)}{O(r)} = \prod_i \lambda_i \quad . \tag{4.11}$$

Thus, the a priori odds of an assertion have no influence on the propagation of confidence, only on the proper interpretation of the final odds-factor.

These two expressions depend only on the odds-factors of assertions and not on any linear or non-linear transformation of them. It is not necessary to convert from odds to probability to compute the non-linearity implied by figure 4.12, for example. This simplification leads to a dramatic savings in the computational cost of propagating

confidence.

Numerical Assertions

The processing of support for numerical (f0) assertions means taking in f0 estimates and combining them to yield a probability density for f0. Each contributing assertion is in the form of a Gaussian estimate for f0 with a specified probability of false alarm. It is assumed that an estimate that is in fact a false alarm contributes nothing to the probability for f0. Thus for two contributors there are four possible cases of false alarms: both true, both false, one or the other false. For n contributors, there are 2^n combinations. The probability of any combination is computed from the vector of probabilities of false alarm by taking the product of the probabilities that the false ones are false times the product of the probabilities that the true ones are true. For each combination, only the true estimates influence the density, those are multiplied. Thus the overall density is given by the sum of weighted products of Gaussians, where the weight placed on each product is the probability of that particular combination of true and false estimates.

$$p(f_0) = \sum_{combos} \left(\prod_{j \in false} P(\bar{j}) \prod_{k \in true} P(k) Gaussian_k(f_0) \right)$$

An incremental form of this computation exists whereby a single new estimate s may be used to convert the density for f0 before (p) to the one after ($\tilde{p}$) as follows:

$$\tilde{p}(f_0) = P(s)p(f_0)Gaussian_s(f_0) + P(\bar{s})p(f_0) ,$$

which allows the cost of computation of probability for numerical values to be linear in the number of supporting assertions.

4.5. Numerical Pitch Detection

The numerical pitch detector used in the PDA is based on temporal similarity. First, a map is made of the local similarity of the waveform in the vicinity of a selected index. Then the glottal pitch is estimated from that map by imposing some constraints on the uniformity of the periods.

The same similarity map that is created in the first stage of numerical pitch estimation is also used to corroborate voicing. For that purpose there is no period uniformity constraint, so the second stage is not needed.

4.5.1. Normalized Local Autocorrelation

We call the procedure for creating a similarity map a "normalized local autocorrelation (NLA)". A sequence $x[k]$ is windowed twice, once at the location n where the pitch is to be estimated, and again somewhere nearby (e.g. at $n+l$). These two windowed sections are each normalized to unit energy and their inner product is the similarity estimate at location n for the lag l. In geometric terms this procedure is measuring the cosine of the angle between the vectors which are the two sections, without regard to the lengths of those vectors.

If we define a sequence of vectors $\vec{x}[s]$ which stand for the input signal shifted to the left by by s, then for position n, the equations for NLA are[2]

$$\vec{y}_1[n] = \vec{x}[n] \times \vec{w} \tag{4.12a}$$

$$\vec{y}_2[n, lag] = \vec{x}[n + lag] \times \vec{w} \tag{4.12b}$$

[2] A "$\vec{\ }$" mark signifies a 1-dimensional sequence. The "$\vec{\ }$" denotes normalization to unit energy. "×" represents element by element multiplication and "$< \vec{x}, \vec{y} >$" means the unweighted multiplicative inner product over the implied indices of the sequences.

$$\underline{\vec{y}}_1[n] = \frac{\vec{y}_1[n]}{\sqrt{<\vec{y}_1[n], \vec{y}_1[n]>}} \tag{4.13a}$$

$$\underline{\vec{y}}_2[n, lag] = \frac{\vec{y}_2[n, lag]}{\sqrt{<\vec{y}_2[n, lag], \vec{y}_2[n, lag]>}} \tag{4.13b}$$

$$s[n, lag] = <\underline{\vec{y}}_1[n], \underline{\vec{y}}_2[n, lag]> \tag{4.14}$$

Amplitude Variation in Speech

In estimating similarity for either period or voicing determination, it is preferable for amplitude not to be a factor. This is because the speech signal can be highly variable in amplitude over regions where the waveshape is relatively constant. Examples of such variation are the growth and decay at voice onset and offset (an example for voice onset is seen in figure 4.13). In a section with a relatively stable waveshape, the amplitude can vary by a factor of 2-5 in as little as 20ms.

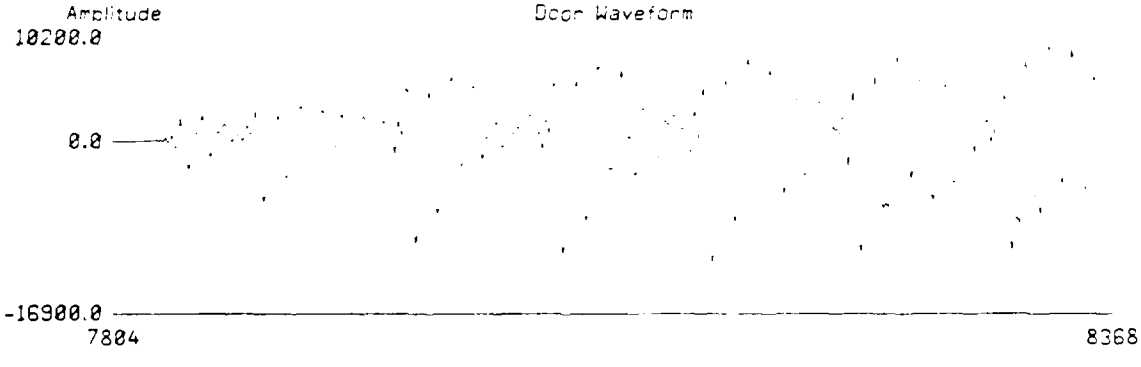


Figure 4.13 Amplitude Variation in Speech

Effects on a Similarity Estimate

Of interest here (for period estimation and voicing corroboration) is the ability to find nearby locations of similar waveshape, since that is what indicates glottal vibration. True periodicity (identical waveshape *and* amplitude) is not necessary.

The effect of amplitude sensitivity in a similarity estimate is to make the peak values of the similarity erratic. This means that interpreting those peaks in absolute terms can lead to errors in voicing judgment. Further, if (as in the PDA) peak information is used to estimate the confidence in the f0 value, then those confidence estimates will also be inappropriately sensitive to amplitude variation.

Reducing Sensitivity

Like short-time autocorrelation[6], NLA analyzes its input signal "locally" to reduce sensitivity to long time variation in waveshape and period. Like modified short-time autocorrelation[6] NLA produces the same value at any lag corresponding to a period. It has no linear taper as does standard autocorrelation. Like any normalized measure (e.g. "semblance" processing[63]) the NLA will produce a value (i.e. 1.0) at $n*period$ that is independent of any scaling of the input signal. What NLA accomplishes that other normalized measures do not is to continue to produce this same value when its input is an exponentially weighted periodic signal instead of a purely periodic one[3]. Thus, similarity estimated with NLA is sensitive to the important phenomenon (waveshape) and insensitive to the interfering effects of growth and decay.

---

[3] This can be understood by recognizing that an exponential envelope on x appears as a simple scaling on delayed versions of x. Since the NLA is immune to scaling on *either* section in the inner product, the impact of exponential weighting is nil.

Warping the Similarity

On voiced speech, NLA values at *n*period* lie between about .7 and 1.0 †. In order to spread this region out and thereby give better control over the setting of similarity thresholds in the system, the warping function

$$y = \log\left|\frac{(1+x)}{(1-x)}\right| \quad \& \quad -7 \leqslant y \leqslant 7 \tag{4.15}$$

was used. This warp spreads the relevant values over the range from about 1 to 7.

In other portions of the PDA system, this similarity value was used as a kind of odds. That is, the similarity was treated as signifying the likelihood of that lag being the period of the speech or a multiple of it. While high similarity signifies high likelihood and low similarity signifies low likelihood, there is no mathematical justification for this warped similarity being treated as odds. We simply chose to treat it as such because it had the right behavior and no obviously better alternative was apparent. In any following text, numerical references to similarity values refer to the warped similarity.

The Choice of Window

The similarity function is a two dimensional quantity $s[n, lag]$. The weighting function $\vec{w}$ serves to localize the estimate to the vicinity of n, and to band-limit the response of $s[n, lag]$ as function of n. From the definition above, if we temporarily ignore the complexity caused by the normalization of $\vec{y}_1[n]$ and $\vec{y}_2[n, lag]$ we can approximate $s[n, lag]$ as follows:

$$s[n, lag] = \sum \vec{x}[\cdot+n] \times \vec{x}[\cdot+n+lag] \times \vec{w}^2 \ . \tag{4.16a}$$

† These values are based on experiments and not on any theoretical properties of NLA or speech.

For a fixed lag this can be rewritten as a function of n

$$s_{lag}[n] = \sum \vec{z}_{lag}[\cdot + n] \times \vec{w}^2 \qquad (4.16b)$$

where

$$\vec{z}_{lag} = \vec{x} \times \vec{x}[\cdot + lag] \ , \qquad (4.16c)$$

or equivalently

$$\vec{s}_{lag} = \vec{z}_{lag} * \vec{w}^2 \ , \qquad (4.16d)$$

where * represents correlation. From this expression we see that the correlation between $\vec{w}^2$ and $\vec{z}$ serves to smooth $\vec{z}$, and thereby band-limit the response of $s[lag, n]$ as a function of n. Further, it is evident that we should choose $\vec{w}$ to be the square root of some suitably band-limiting window (e.g. Hamming).

The choice of bandwidth for this window depends on the stability of the waveform similarity with n and the sampling interval over n. In the PDA, the waveform similarity is computed every 10ms (a value that is common to most pitch processing), if that is a sufficient sampling density, it suggests that the similarity variation as a function of n should have no spectral components above 50 Hz.

A 25 ms Hamming window has substantial response up to 40 Hz and little above this frequency so that should pass the relevent part of the similarity variation while suppressing irrelevent higher frequency components. Experiments we performed demonstrated that shortening the window to 12.5 ms lead to unacceptably low stability for the similarity estimate.

Implementation

The expressions (4.12a) through (4.14) present a direct implementation of the normalized local autocorrelation. A faster FFT based implementation of the complete

NLA algorithm is shown in figure 4.14. The dominant computations in this implementation are the two correlations (*'s) each requiring two forward and one reverse FFT. Assume an FFT size m of 2048 points (large enough to capture the 1200 lags desired) and $\frac{m}{2}\log_2(\frac{m}{2})$ butterflies per FFT. Since a butterfly takes one complex multiply and two complex adds (4 real multiplies and 6 real adds), the total arithmetic operation count is roughly $10\times1024\times10 = 100{,}000$. Using this implementation kept the per sentence processing time (for NLA) to about 17 minutes. Because the processing time was independent of the window length (up to about 80ms), the window length was solely determined by the tradeoff of temporal stability for temporal resolution.

4.5.2. Determining Pitch from Similarity

Given a similarity map generated with warped NLA, several factors need to be considered in estimating pitch.

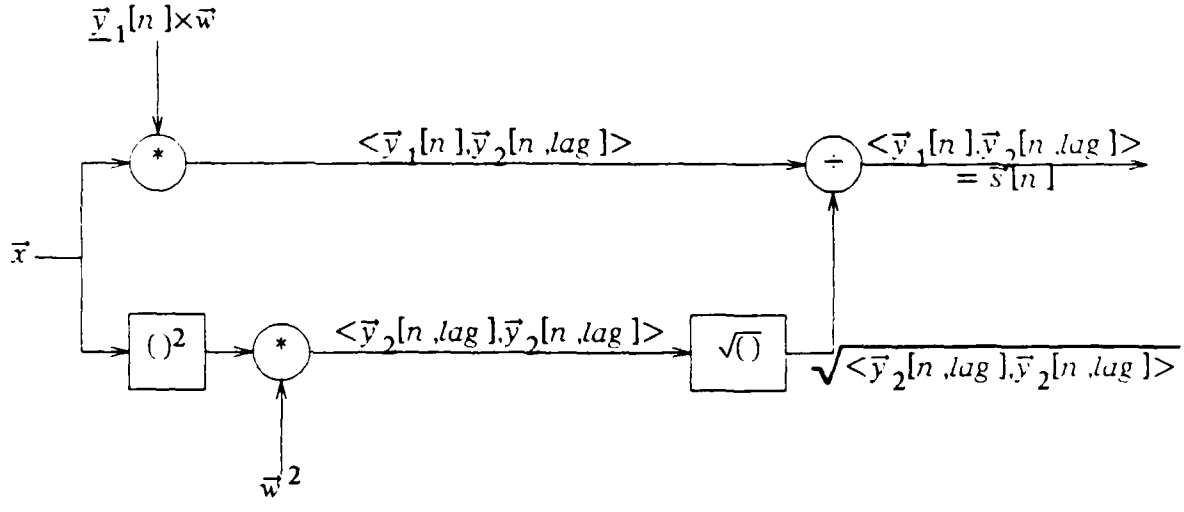


Figure 4.14 Fast NLA

1 Speech has period jitter of at least .5%.

2 Periodic voiced regions of a sentence can have moveout (monotonic period change) of several percent per cycle.

3 Waveform cycles excited by glottal pulses are typically present in groups of three or more.

4 Speech does not typically have strong similarity within a period.

5 Above a few kHz, speech similarity becomes erratic.

These thoughts led us to the following procedure for estimating pitch from the similarity map.

a Select a candidate similarity peak in the lag range from 2 ms to 20 ms whose similarity is at least 1.0.

b Search for corroborating peaks near 2 times and 3 times the lag of the candidate.

c Starting with a net odds that is the similarity of the candidate, if the similarity of the 2x peak is >1.0 then multiply the net odds by it. If the similarity of the 3x peak is *also* >1.0 then multiply the net odds by it also. Thus the net odds is given by

$$net\_odds = sim(peak_{1x}) \tag{4.17}$$

if $sim(peak_{2x}) < 1$, or

$$net\_odds = sim(peak_{1x})\,sim(peak_{2x})\,max(1.0, sim(peak_{3x})) \ , \tag{4.18}$$

if $sim(peak_{2x}) \geq 1$.

These steps generate two sets of peak candidates, one for positive lags and one for negative lags. Positive and negative candidate sets are processed separately from one another in two passes through the following step.

d For each candidate in the set, look for others that are at lags nearer to the origin. If found, the largest similarity of those nearer peaks is divided into the net odds of the candidate.

At this point one direction is selected and the other discarded. Only one direction is kept because the f0 assertions that would result from looking in both directions would

be very correlated. This would violate the assumption of the knowledge manager that multiple assertions on a given topic can be taken as independent information on that topic.

e Choose the lag direction (positive or negative) that includes the candidate with the best net odds. Discard the candidates in the other direction.

f Make each remaining candidate with a net-odds > 1.0 into a Gaussian f0 assertion at that speech index. The mean f0 of the assertion is $\frac{1}{P_0}$ for that candidate (see below), the odds of the assertion is given by the net-odds of the candidate, and the standard deviation is given by an uncertainty estimation formula given below.

Speech properties 1-5 are handled by this procedure as follows:

1,2 In step b, corroborating peaks must come from the vicinity of 2 times and 3 times the candidate lag, but not exactly at those positions. The tolerance is +−2% about the expected position. This allows for both period jitter in the individual cycles and any consistent moveout between the first second and third cycles.

The support that the multiples give to the candidate is based on how close they are to their expected position. A triangular weighting is applied to the support given by the multiple. The weighting is 1.0 if it is exactly where expected and 0.0 if it is at or exceeds +−2% deviation from the expected position.

3 By searching for multiples of a candidate lag, the procedure uses the existence of multiples to increase its confidence in the candidate. Lack of such multiples is *not* taken as refutation of the candidate, but such a candidate would need to be very strong to stand by itself against other candidates with multiple support.

4 This knowledge is dealt with in step d. The presence of a strong similarity inside a candidate lag reduces the net odds for that candidate by the similarity of the strongest inner peak. Thus, other things being equal, the shorter of two candidate choices will retain more confidence because the net-odds of the longer will be divided by the primary peak similarity of the shorter.

5 The fact that the speech spectrum above 2 kHz does not carry much similarity information was established by experiment. This effect was also noted by a co-worker who was attempting to track the phase of upper harmonics of voiced speech for coding purposes[64]. Since poorly correlated upper harmonics (and high frequency noise) will suppress peaks in the similarity map, the speech is filtered to eliminate frequencies above 2 kHz.

The above description of the numerical pitch detector applies when it is running without prior information about the sentence. In the PDA, the numerical pitch detector is given prior information about certain portions of the waveform. The next section discusses how this is accomplished.

4.5.2.1. Numerical Pitch Advice

In some regions of the sentence, the phonetic context has implications for pitch detection. Information about either the expected temporal behavior of f0, or about how to go about detecting it. In these regions, the numerical pitch detector is given phonetically derived "advice" which takes four forms.

I The moveout can be specified.
II The preferred direction can be specified.
III The expected range of pitch values can be specified.

Moveout

In certain phonetic contexts the pitch is expected to change in a consistent fashion from period to period. For example, in a vowel-consonant-vowel context where the consonant is /p/, /t/, or /k/, we expect a falling f0 at voicing onset. In such contexts, the PDA invokes the numerical pitch estimator with advice the periods should be growing in that region (the growth is estimated as 1% per period as determined by informal experiments).

This advice affects the behavior of the numerical pitch detector in step b. Instead of looking for period multiples at exactly 2 times or 3 times the lag of the primary candidate, an adjustment is made in the expected positions of those multiples. The theoretical basis of that adjustment is as follows.

Suppose we assume that the speech signal is a repeating signal with a linearly changing period. Observations are time values that correspond to known cycles of this repetition (e.g. cycle 1 happened at 1.04, cycle 3 happened at 1.54). Let these "cycle times" be the values of an indexed set of variables $\{\cdots t_{-2}, t_{-1}, t_0, \cdots\}$ and let us assign the time of the zeroth cycle (t_0) to be 0. Let the variable c stand for a continuous "cycle position" parameter for the speech signal. This variable takes on integer values at the corresponding cycles of the waveform (e.g. $c=1$ at cycle 1 and so on.

The following expression defines the value of time at any cycle of the waveform:

$$t_i = \int_0^i \frac{dt}{dc} dc \tag{4.19}$$

If we define the "instantaneous" period of the signal ($P()$) as the rate of change of time with cycle $\frac{dt}{dc}$ then we have

$$t_i = \int_0^i P(c)\,dc \quad . \tag{4.20}$$

Finally, if we require that the period is a linear function of the cycle then

$$P(c) = P_0 + mc \tag{4.21}$$

and

$$t_i = \int_0^i (P_0 + mc)\,dc = P_0 c + m\frac{c^2}{2}\Big|_0^i = P_0 i + m\frac{i^2}{2} \quad . \tag{4.22}$$

Since we have a set of times $\{t_i\}$ we can use equation (4.22) to determine P_0 and m. For example, if we knew $\{t_{-3}, \ldots, t_3\}$ then we would have

$$\begin{bmatrix} \cdots & \cdots \\ 3 & 4.5 \\ 2 & 2.0 \\ 1 & .5 \\ -1 & .5 \\ -2 & 2.0 \\ -3 & 4.5 \\ \cdots & \cdots \end{bmatrix} \begin{bmatrix} p_0 \\ m \end{bmatrix} = \begin{bmatrix} \cdots \\ t_3 \\ t_2 \\ t_1 \\ t_{-1} \\ t_{-2} \\ t_{-3} \\ \cdots \end{bmatrix} . \tag{4.23}$$

In the numerical pitch estimation process candidates are generated together with peaks at the second and third multiple. Equation (4.22) determines where such peaks will be located and given peaks, how to estimate the instantaneous period at $c=0$. Specifically, if there is an a priori estimate of non-zero moveout and the time of the first multiple (the primary candidate) is known, then the expected times for the second and third multiples are given by

$$t_i = t_1 \left| \frac{i + \frac{mi^2}{2}}{1 + \frac{m}{2}} \right| . \tag{4.24}$$

This expression is used to determine the center of the search region used to find these peaks. A correct estimate of the moveout advice will yield much stronger candidates, because the expected positions of the multiples will correspond more closely to their true positions, and that will mean the triangular weighting that is applied to such multiples will not reduce their support as much.

Despite the availability of t_1, t_2 and t_3, the PDA only uses t_1 and t_2 to estimate the period based on a given candidate. This saves having to solve an overconstrained problem, and places less of a burden on the assumption that period is linearly varying with waveform cycle since only two (and not three) cycles are considered. These two advantages are offset by the fact that the estimate of P_0 is potentially noisier than it

might be if all three time values were used.

Preferred Direction

In some phonetic contexts there should be a stronger similarity in one time direction than the other. Let "left" stand for travelling back in time and "right"stand for travelling ahead. At the closure of a /b/ stop, the waveform will have one shape to the right (where the mouth is closed) and another to the left (where it is open). An example of this effect can be seen in figure figure 4.15. A rule which detects this and similar contexts invokes the numerical pitch detector twice, once on each side of this boundary advising each invocation to look for similarity moving away from the boundary and not towards it. The effect of this advice on the numerical pitch detector is to penalize the net odds of candidates that are in the conflicting direction by 50%. The intent of the penalty is to influence the detector to use the candidates in the direction that is more likely to be the correct one.

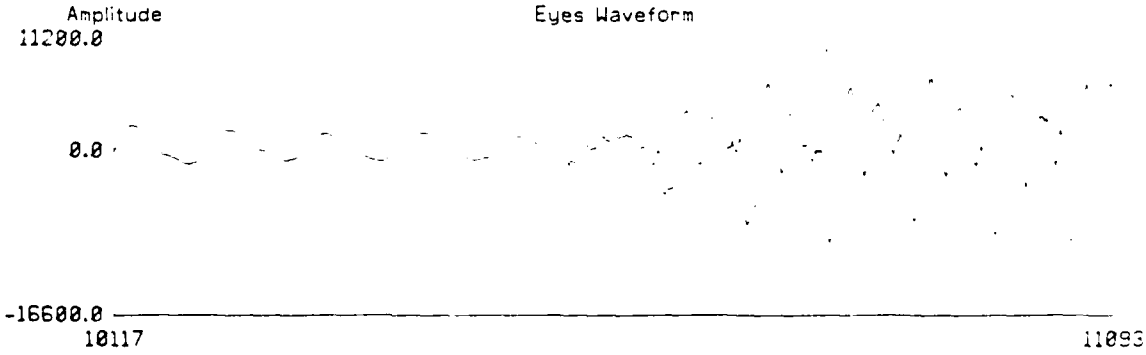


Figure 4.15 Voiced stop closure

Pitch Range

Within the PDA, there are phonetically derived pitch estimates which are based on research results on the average pitch of certain phonemes as spoken by speakers of certain sex/age classes There are also pitch estimates that are the result of the numerical pitch detector. Both of these types of estimates are integrated by the knowledge manager to provide the final results of the system. As is discussed below, such pitch estimates must be given as Gaussian densities for the fundamental frequency with false alarm specification in order for the knowledge manager to be able to integrate them.

Late in the implementation of this system, another kind of knowledge was added to the system concerning the rough pitch range of the sentence. This information was generated by a preliminary numerical pitch analysis that only used data that the detector was very sure of. These pitch estimates generated a histogram of pitch values that was used to estimate the overall f0 range of the sentence. Experience demonstrated that this range gave a tight upper bound on the likely f0 values of the sentence, but that the lower bound was loose owing to the extremely low values of f0 that can result from irregular glottal vibration.

Because the upper bound was tight and the lower bound was not, it did not seem that a Gaussian probability shape was appropriate. Since, the knowledge manager only supported assertions with Gaussian probabilities, this "prior" range estimate was given as advice to the numerical pitch detector rather than as f0 assertions.

This advice takes the form of a sequence of functions which specify the a priori odds on f0 for each frame of the speech. These functions influence the numerical pitch detector in step c. Instead of using the similarity of the primary candidate as the

initial net odds, that similarity is first adjusted by the a priori odds of the candidate.

Jitter

It is also possible to specify the acceptable jitter as advice to this numerical pitch detector. We did not make use of this capability in the PDA.

4.5.2.2. Determining the Standard Deviation of F0 Estimates

Because the f0 assertions generated by the numerical pitch detector must be combined by the knowledge manager (KM) with other f0 assertions, they are given in the format expected by the KM. Each assertion specifies a Gaussian density with an odds at a specific speech index. The knowledge manager combines these densities to generate the composite probability density that is the PDA's statement about f0 at that speech index. The mean of the Gaussian is given by the formula for P_0 given in the previous section. The odds of the Gaussian is given by the net odds of the candidate responsible for the f0 assertion. The standard deviation is estimated by a formula drawn from radar theory[65].

The formula is for the standard deviation of a return time estimate for a radar pulse of a known bandwidth with a known signal-to-noise ratio. The correlation of the transmitted radar pulse with the received one is equivalent to the correlation of the 0 lag speech section with sections taken at various nearby lags. The formula for the variance of the lag estimate is

$$var\,(\hat{\tau}-\tau) = \frac{1}{\frac{2\bar{E}_r}{N_0}\left|\frac{\bar{E}_r}{\bar{E}_r}+N_0\right|\overline{\omega^2}} \qquad (4.25)$$

where

$\bar{E}_r$ *is the signal energy,*

N_0 *is the noise variance, and*

$$\overline{\omega^2} \equiv \frac{\frac{1}{2\pi}\int \omega^2 |S(\omega)|^2 d\omega}{\frac{1}{2\pi}\int |S(\omega)|^2 d\omega} \tag{4.26}$$

is the standard signal bandwidth. By measuring the standard bandwidth $\overline{\omega^2}$, the noise level (in silent regions) and the signal power, the PDA can use equation (4.25) to estimate the standard deviation in the f0 assertions it produces.

4.6. Conclusions

This chapter has described some of the interesting features of the implementation of the PDA:

- A signal representation based on ∞ domains that permits the representation of a broad class of signals including exponentials and periodic signals, and allows the representation of temporal phonemena such as delay and linear phase in a straightforward manner.
- A computational representation for temporal events that merges them based on numerical and symbolic and compatibility, and serves as a means of combining sources of information about temporally distributed phenomena.
- A rule system that facilitates program development and interaction through history independence.
- Rule conditions based on procedures rather than patterns which isolate rules from object data structure to simplify programming, and make efficient use of the information encoded by linkage in the assertion database.
- An efficient algorithm for rule condition procedure evaluation.
- A new representation for symbolic assertion confidence that allows for temporal variation and thereby represents time varying problems more accurately.
- A new method for computing the confidence of symbolic assertions (using odds-factors) that eliminates the need for approximating the Bayesian analysis and is substantially less expensive.
- The NLA, a new periodicity measure that is insensitive to growth and decay in the waveform.

CHAPTER 5

System Evaluation

5.1. Introduction

This chapter describes a pilot experiment to determine the performance of the PDA. In it, the PDA and the Gold Rabiner (G-R) pitch detector are compared, using noisy speech with a hand edited reference pitch track.

We chose to do this experiment in order to determine if the idea of combining numerical and symbolic processing in this program was beneficial. Despite the modest amount of symbolic and numeric knowledge that had thus far been incorporated into the PDA[1], the experiment demonstrated that the PDA produced 1/2 as many errors as G-R accross many different signal-to-noise ratios.

To determine the contribution of symbolic information to the PDA's performance, the system was run on two of the test sentences without the aid of a phonetic transcript or the sex/age of the speaker. Without this symbolic information, the PDA and the G-R pitch detector performed equally well in terms of total voicing and f0 errors. Surprisingly the symbolic information impacted most on f0 errors and not voicing errors. Apparently, the addition of relatively broad f0 assertions due to the symbolic information is an appropriate complement to the precise, but multi-modal, f0 densities produced by numeric analysis.

---

[1] Many symbolic ideas (such as tune information) and numeric ideas (such as interframe f0 smoothing) had not been incorporated.

5.2. A Comparison of PDA and G-R

This is a three way comparison between the PDA, the Gold-Rabiner pitch detector (G-R) and hand marked pitch. The G-R program was implemented on a lisp machine using information derived from the original paper that described it[3], and information drawn from a C language program[2] written by Eliot Singer (a member of the group at MIT Lincoln Laboratory in which Ben Gold works). The sentences (already hand marked for pitch every 100 ms) were acquired from Bruce Secrest who was at the time working for the speech research group at Texas Instruments.

The frame rates for both detectors were 100/sec, the speech was sampled at 12.5 kHz and low-passed below 900 Hz before analysis. Fourteen sentences were used, including 10 different texts, male and female speakers, and age categories ranging from child to elderly. Each of the fourteen sentences was used at 5 signal energy to noise energy ratios (40 dB, 30 dB, 20 dB, 10 dB, 0 dB) by adding white Gaussian noise.

5.2.1. Performance Criteria

Four kinds of errors were counted: high and low pitch errors of more than 30 Hz, voiced-to-unvoiced errors and unvoiced-to-voiced errors. The errors were divided by the number of seconds in each sentence. Since the frame rate was a uniform 100 frames/second, these numbers may be interpreted both as percentage errors (100 times the number of errors per frame), and as error rates (errors per second).

These errors were further gated by an amplitude criterion. Specifically, if the speech amplitude in the frame in question never exceeded 1/10 of the maximum amplitude of the sentence then the frame was ignored. This deemphasizes the

[2] Knowledge embedded in that C program that either differed from the original paper (e.g. voting matrix bias values) or was not present in the original paper (e.g. the precise speech filter spectrum) was essential to achieving the best performance from the detector.

significance of errors in the quiet portions of the sentence, errors that would not be likely to significantly affect the quality of speech reconstruction.

The standard deviation of f0 "error-free" voiced frames was also measured. However, as is explained below, the hand marked pitch was too imprecise for this figure to be meaningful.

5.2.2. Caveats

The hand marked pitch tracks were only measured to the nearest sample lag (as evidenced by the quantization of f0 values). This means that precise f0 deviation measurements cannot be made with this data since they would be dominated by the f0 quantization. We have presented these deviation measurements, but do not include them in our criteria for performance.

It appears that peak-picking in the time domain was used to hand mark the pitch. We believe this because in places where there was f0 ambiguity between the apparent pitch on the basis of peak-picking and other plausible techniques such as correlation measurements (as was used in the semi-automatic system SAPD[66]) it was the peak-picking value that was chosen.

If peak-picking was the sole method used to estimate f0, then the hand marking may not be the best possible. However, there is no reason to believe that it would give a preference to either of the pitch detection methods being tested, and a cursory examination of the hand marking did not reveal extensive anomalies.

Another factor that might influence the results is our use of an amplitude gating on the errors. It seems clear that there should be some amplitude weighting, since errors made at low amplitude would not be as noticeable to someone listening to

reconstructed speech. Yet there is no standard for such a weighting. If our particular weighting isn't ideal, it is probably close enough that the effects of the difference on this experiment are minor.

The performance of the PDA is enhanced by the fact that the transcript is unaffected by the noise. It would have been fair to "add noise" to the transcripts as well. However, there was insufficient time to retranscribe the utterances after noise was added. Therefore, admitting to the potential bias, the transcript (and sex/age) information used in these experiments is the information derived from clean sentences.

As has been mentioned, our primary interest was in learning methods of combining knowledge, symbolic and numeric, and not in optimizing the PDA. Thus we did not incorporate all the knowledge that might have improved the performance of the PDA.

In building the PDA, little effort was expended adjusting thresholds, window sizes and shapes, filters etc. Reasonable values were chosen based on intuition, or brief experiments with one or two sentences. Though no experiments on the sensitivity of these parameters were performed, it is unlikely that they are all near their optimal values.

Conversely, these sorts of refinements are apparent in the Lincoln Laboratory version of the G-R pitch detector, and minor deviations from the parameter values in that program led to markedly poorer performance. Thus it seems likely that the G-R program is the best example of a program in its class, whereas the PDA is only a "rough cut".

While the following tests are hardly conclusive (owing to the limited corpus and the caveats above) they suggest that the combination of symbolic and numeric ideas

can lead to better performance than either taken alone.

5.2.3. Test Results

An example taken from the raw data that was gathered is shown in table 5.1. These were the 5 different trials (different SNR's) for a single sentence. The sentence was spoken by speaker "NLD" using text 9 - "Almost everything involved making the child mind.". The speaker was female and in age group 2 - adolescent (we were not able to get specific age information). The other columns in the figure are:

SNR — The signal to noise ratio (in dB) used for that trial.

High — Errors in estimated f0 that exceeded the hand marked value by at least 30 Hz.

Low — Errors in estimated f0 that fell short of the hand marked value by at least 30 Hz.

UV-V — Errors in which the speech was marked "voiced" when it should have been marked "unvoiced".

V-UV — Errors in which the speech was marked "unvoiced" when it should have been marked "voiced".

Dev — The standard deviation of f0 weighted by the normalized amplitude of speech in each frame.

	Test				PDA					GR				
Spkr	Txt	Sex	Age	SNR	High	Low	UV-V	V-UV	Dev	High	Low	UV-V	V-UV	Dev
nld	9	f	2	0	0.4	4.0	0.0	0.0	2.1	0.0	9.1	0.0	11.5	2.5
nld	9	f	2	10	0.7	0.9	0.0	0.0	1.7	0.2	2.0	0.4	3.1	1.7
nld	9	f	2	20	0.7	0.9	0.0	0.0	1.7	0.2	0.4	0.0	2.0	1.5
nld	9	f	2	30	0.4	0.7	0.0	0.4	1.6	0.4	0.4	0.0	2.2	1.5
nld	9	f	2	40	0.7	0.7	0.0	0.0	1.6	0.4	0.4	0.0	2.0	1.6

Table 5.1 Example of Results

As was mentioned above, the standard deviation values are of little value because the reference f0 was so heavily quantized. Therefore, to estimate the overall performance of the two methods at each SNR, the error columns were added and averaged across all sentences with the same SNR. Those results are shown in table 5.2. The columns marked "F0" are the total gross f0 errors, and the columns marked "Vcng" are the total voicing errors. The parenthetical numbers in this table are the standard deviations of the numbers above them. In interpreting these tables, differences of about 1.0 errors per second or less should be considered barely significant.

This experiment indicates that the PDA makes roughly half the total errors (sum of f0 and voicing errors) of G-R irrespective of SNR, and that the difference is largely due to the V-UV errors in G-R. While it might seem nonsensical that G-R is making fewer high f0 errors when the SNR is low, in reality G-R is just calling those frames unvoiced, leading to a reduction in f0 errors at the expense of V-UV errors. Because of this sort of effect, the behavior shown by individual columns of this table must not be taken out of context.

	PDA						GR					
SNR	High	Low	UV-V	V-UV	F0	Vcng	High	Low	UV-V	V-UV	F0	Vcng
0:	1.4	4.1	1.4	0.1	5.5	1.5	0.3	4.3	0.1	13.6	4.6	13.7
	(1.1)	(2.7)	(2.4)	(0.2)	(2.1)	(1.7)	(0.4)	(4.3)	(0.3)	(5.2)	(3.0)	(3.7)
10:	1.4	0.9	0.2	0.7	2.4	0.9	0.8	1.5	0.2	5.4	2.3	5.7
	(1.5)	(0.9)	(0.5)	(0.7)	(1.2)	(0.6)	(0.8)	(1.3)	(0.4)	(3.0)	(1.1)	(2.1)
20:	1.4	0.8	0.2	0.9	2.1	1.1	1.0	1.1	0.2	3.4	2.1	3.6
	(1.6)	(0.8)	(0.2)	(0.8)	(1.3)	(0.6)	(0.6)	(1.3)	(0.3)	(2.1)	(1.0)	(1.5)
30:	1.5	0.8	0.3	0.2	2.3	0.4	0.9	0.9	0.1	3.7	1.9	3.8
	(1.6)	(0.8)	(0.4)	(0.2)	(1.3)	(0.3)	(0.6)	(1.0)	(0.2)	(2.3)	(0.8)	(1.7)
40:	1.6	0.8	0.3	0.1	2.4	0.4	0.9	0.9	0.1	3.9	1.8	4.0
	(1.7)	(0.8)	(0.4)	(0.2)	(1.3)	(0.3)	(0.5)	(1.2)	(0.2)	(2.9)	(0.9)	(2.0)

Table 5.2 Performance vs SNR

This effect also points out a significant difference in the approach of these two systems. The G-R pitch detector attempts to find periodicity; if it can't G-R calls the speech unvoiced. The PDA determines if the speech is voiced separately and without regard to periodicity. Having determined that the speech is voiced, the PDA chooses the most likely pitch. This means that when the analysis becomes difficult, G-R will begin making large numbers of V->UV errors, whereas the PDA will have a mixture of voicing and f0 errors.

5.2.4. A Typical PDA Run

Figures 5.1a through 5.1f show the results of typical PDA runs from 40 dB SNR

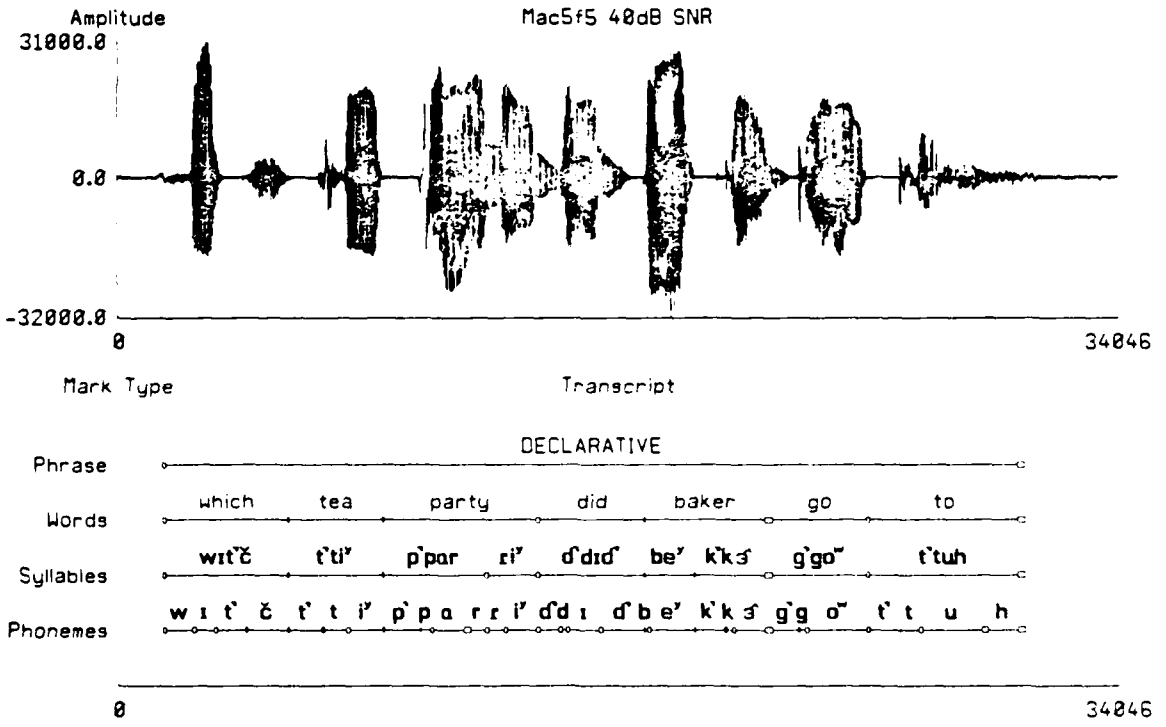


Figure 5.1a Input Information

to 0 dB SNR. The first figure shows the input information for this sequence of figures (the waveform and transcript). The remaining figures show the outputs for different SNR's. The four panels of each figure (from the top) are the voicing assertions, the probability of VOICED, the pitch track and the f0 probability density. Values for the pitch track that are at the top of the graph signify an unvoiced decision.

Some significant features of these runs are

- Below 30 dB SNR, the PDA is unable to locate numeric silence regions reliably. This can be seen as a loss of "ns" marked assertions in the voicing assertions panels of the figures.
- By 0 dB SNR the PDA can no longer numerically resolve distinct voiced intervals. At 0 dB, there is only a single numeric voiced assertion.
- As the SNR drops one can see an increase in the PDA's uncertainty of f0. This takes for form of a thickening of the contours for f0 in the density plot.
- Likewise, there is an overall drop in the voicing certainty as can be seen by noting the drop in vertical scale factor for the probability of "voiced" as SNR goes down.

5.2.5. The Impact of Symbols

In order to judge the impact of the symbolic information on the performance of the PDA, one of the sentences was reprocessed at all 5 SNR's without the input of a transcript or the sex/age of the speaker. The first sentence shown is the one from table 5.1, which had relatively few errors for either detector. The results are shown in table 5.3. The only significant change is a 5/1 increase in gross low f0 errors at 0 dB snr. Note also that the data for the G-R detector corresponds closely with the previous data despite the use of a different noise sequence (the seed for the noise generator was different). This indicates that the experimental results are not extremely sensitive to the specific noise sequence.

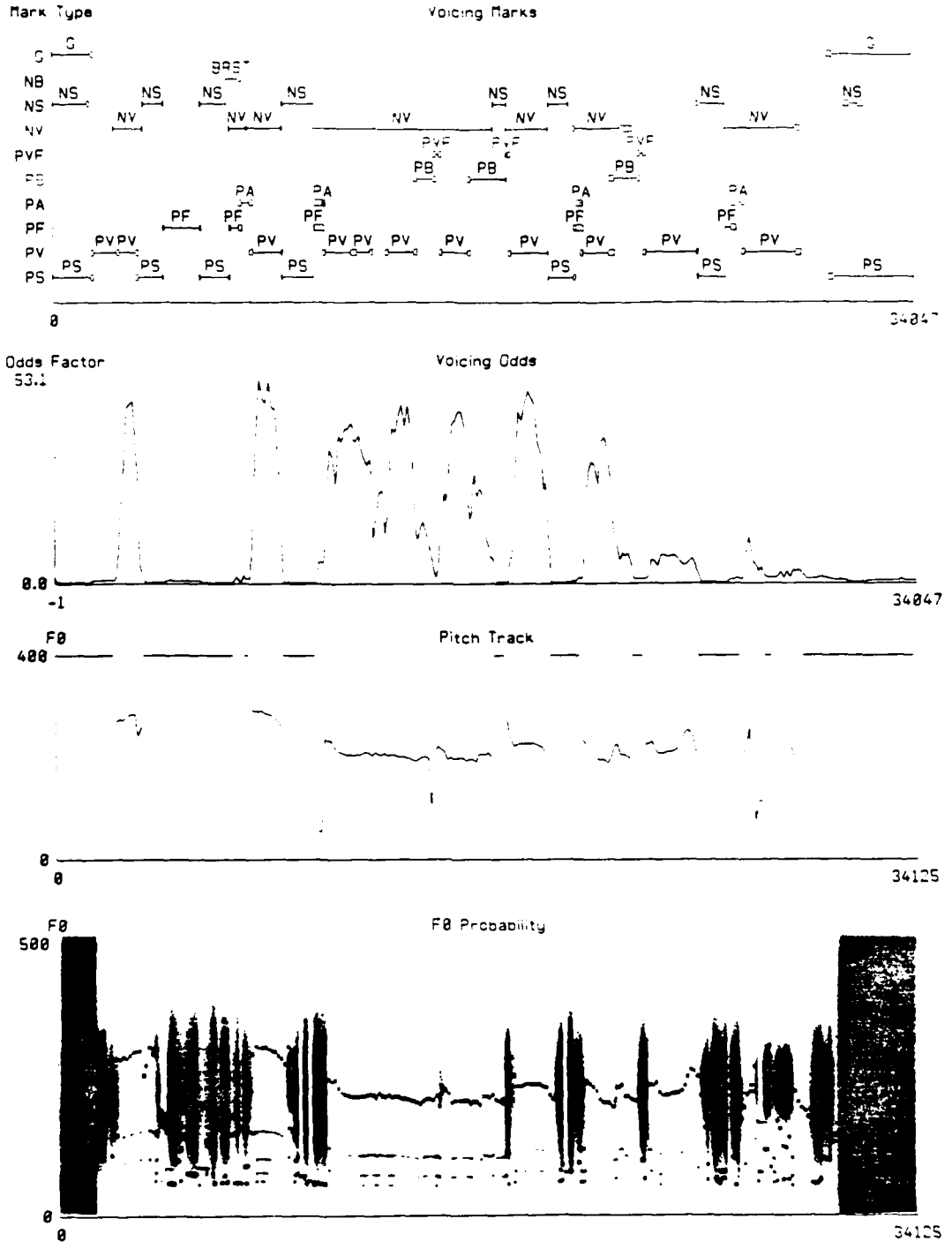


Figure 5.1b PDA Outputs at 40dB SNR

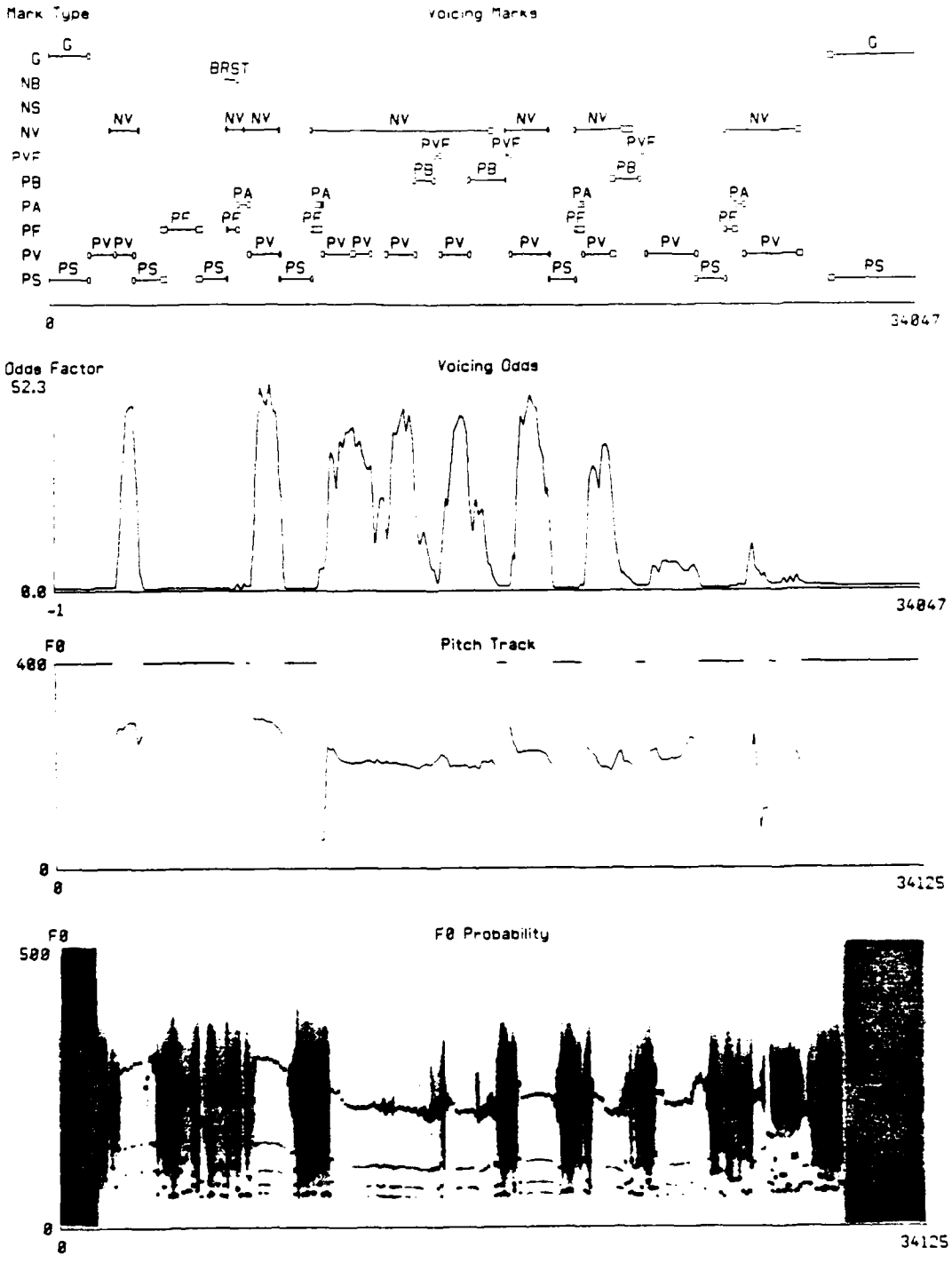


Figure 5.1c PDA Outputs at 30dB SNR

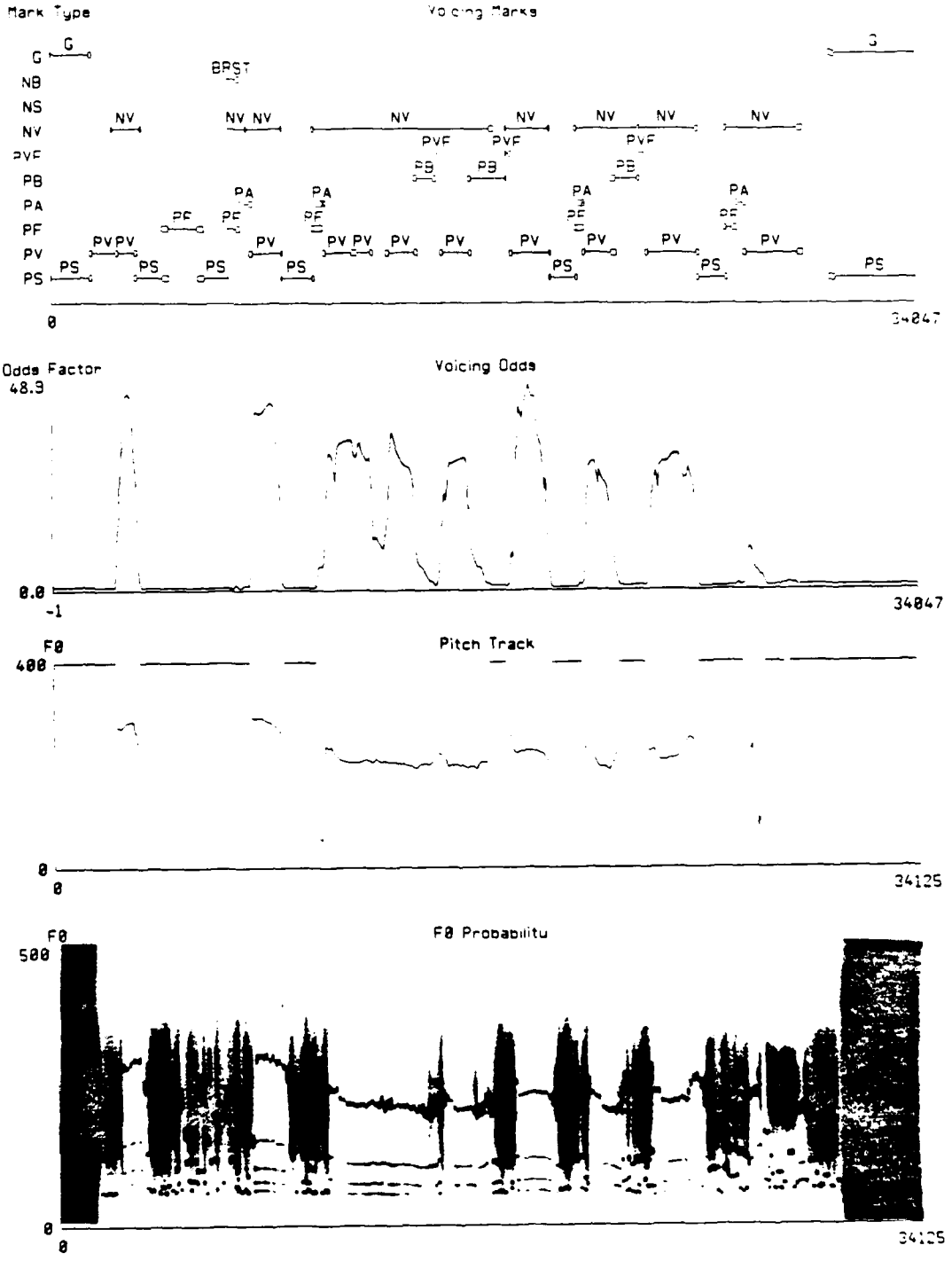


Figure 5.1d PDA Outputs at 20dB SNR

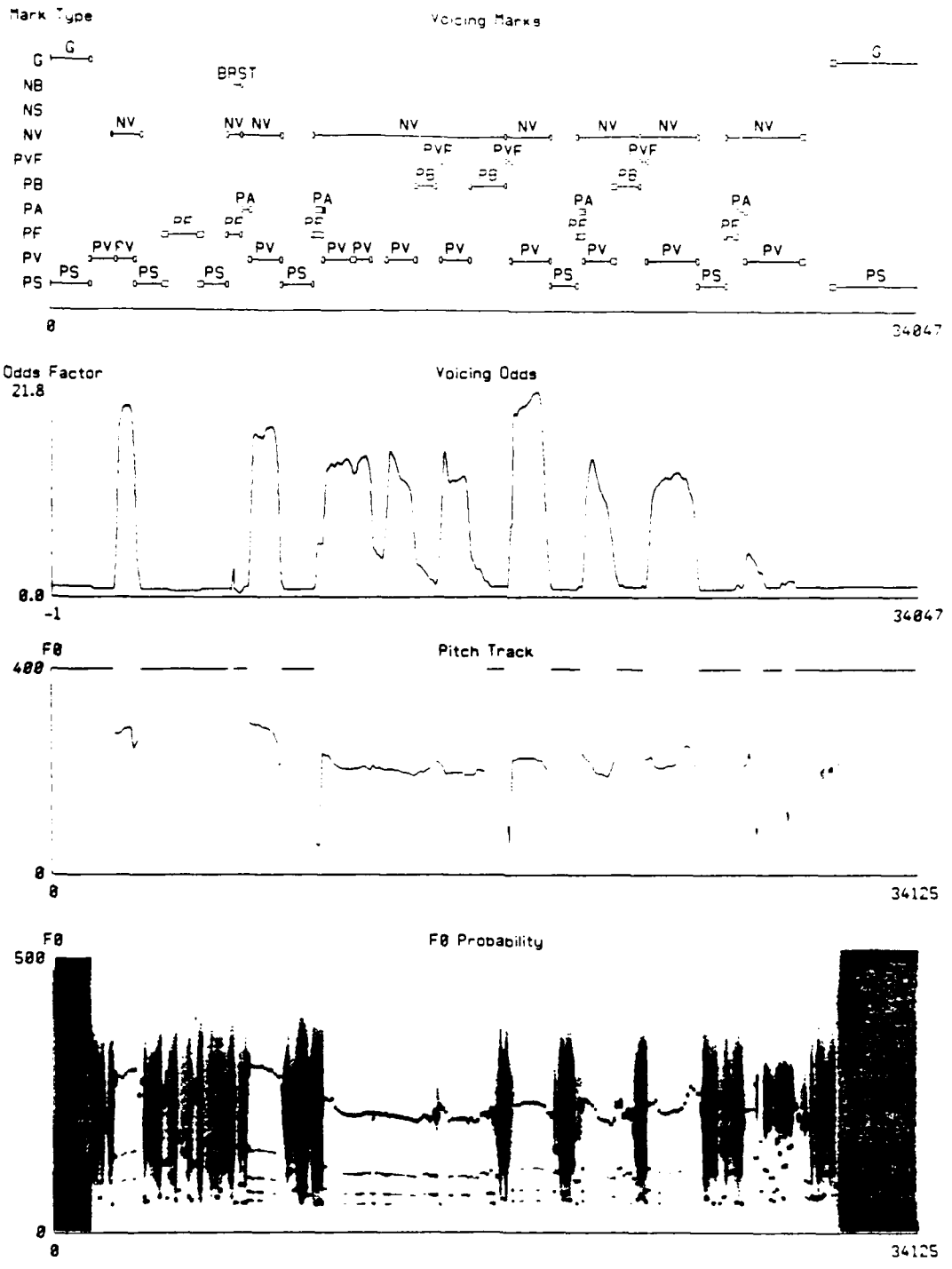


Figure 5.1e PDA Outputs at 10dB SNR

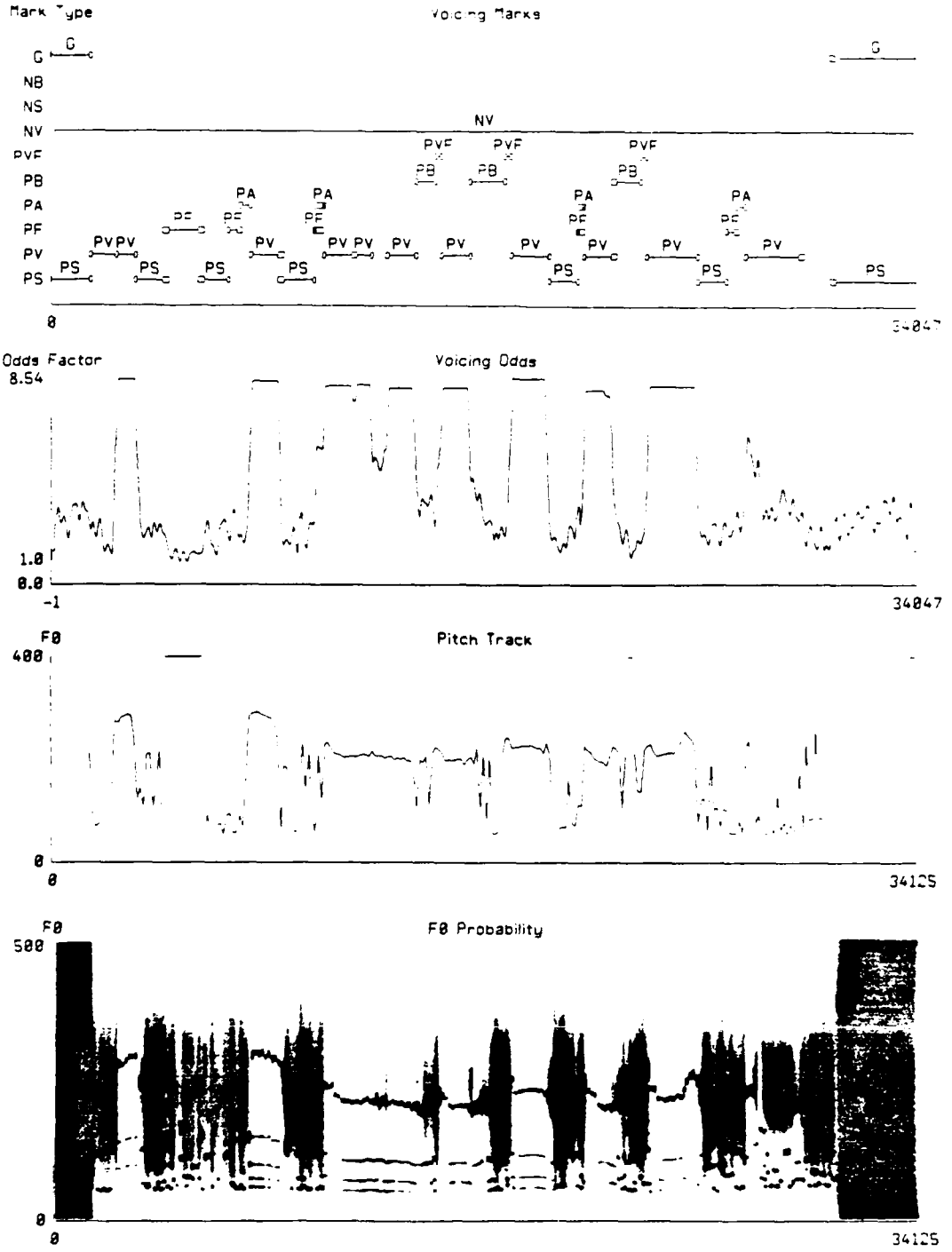


Figure 5.1f PDA Outputs at 0dB SNR

	Test				PDA					GR				
Spkr	Txt	Sex	Age	SNR	High	Low	UV-V	V-UV	Dev	High	Low	UV-V	V-UV	Dev
nld	9	f	2	0	0.2	20.2	0.7	0.2	2.2	0.0	11.1	0.2	10.6	2.7
nld	9	f	2	10	0.4	0.9	0.2	0.4	1.8	0.2	2.4	0.0	2.4	1.4
nld	9	f	2	20	0.4	0.9	0.2	0.2	1.7	0.4	0.4	0.2	2.2	1.6
nld	9	f	2	30	0.2	0.9	0.2	0.4	1.6	0.2	0.4	0.0	2.0	1.6
nld	9	f	2	40	0.4	0.9	0.2	0.2	1.6	0.2	0.4	0.0	2.0	1.6

Table 5.3 Processing without Symbols

The error rates for that sentence were quite low, and one wouldn't expect much impact from the symbols on such clean numeric data. Tables 5.4a and 5.4b show the

	Test				PDA					GR				
Spkr	Txt	Sex	Age	SNR	High	Low	UV-V	V-UV	Dev	High	Low	UV-V	V-UV	Dev
les	2	f	5	0	2.1	5.8	1.3	0.0	2.4	1.1	1.8	0.0	14.8	2.9
les	2	f	5	10	1.6	1.6	0.0	1.8	2.2	0.8	0.0	0.3	6.6	2.8
les	2	f	5	20	1.3	1.8	0.0	1.8	2.3	1.3	1.6	0.8	3.4	2.0
les	2	f	5	30	1.1	1.6	0.3	0.5	2.3	2.1	0.0	0.3	3.2	2.6
les	2	f	5	40	1.3	1.6	0.8	0.5	2.3	1.3	0.0	0.5	3.4	2.5

Table 5.4a Difficult Sentence with Symbols

	Test				PDA					GR				
Spkr	Txt	Sex	Age	SNR	High	Low	UV-V	V-UV	Dev	High	Low	UV-V	V-UV	Dev
les	2	f	5	0	0.3	12.1	1.8	0.8	2.2	1.1	2.6	0.0	12.7	2.3
les	2	f	5	10	0.8	5.0	0.0	2.1	1.9	0.8	2.1	0.0	7.7	2.5
les	2	f	5	20	0.8	3.7	0.0	2.1	2.1	1.0	1.3	0.8	4.7	1.9
les	2	f	5	30	0.5	4.0	0.3	0.8	2.0	1.6	0.0	0.3	2.9	2.2
les	2	f	5	40	0.8	4.0	1.1	0.8	2.0	1.6	0.0	0.5	3.4	2.3

Table 5.4b Difficult Sentence without Symbols

impact of symbols on a more difficult sentence. There is a substantial reduction in the gross low f0 errors at the expense of a slight increase in gross high f0 errors. The total number of gross f0 errors at 0 dB SNR dropped from 12.4 per second to 7.9 per second. At 40 dB SNR the reduction was from 4.8 per second to 2.9 per second. Surprisingly, the symbols appear to have little or no effect on voicing errors. Apparently, the numerical methods for determining voicing in the PDA are more robust than the numerical methods for determining f0.

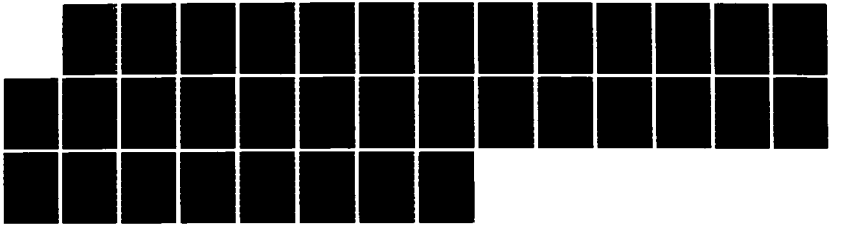
5.2.6. Comments

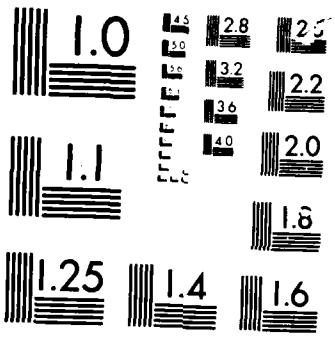
A comparison between the PDA and G-R showed that the PDA made 1/2 the number of errors. The fact that this improvement in performance was accomplished with a program that only partially taps the available knowledge and lacks extensive polishing, indicates that more substantial gains can be made. More significantly, it is clear that the symbolic input is an important contributor to the PDA performance.

Without symbolic input, the two methods have roughly equivalent error rates with the PDA being dominated by low f0 errors, and G-R being dominated by V-UV errors. This difference in the nature of their errors is not surprising, since G-R will never accept the irregularly spaced pitch pulses of glottalization as voiced, whereas PDA can do so since its methods of determining voicing are not predicated on periodicity.

AD-A169 869 KNOWLEDGE-BASED PITCH DETECTION(U) MASSACHUSETTS INST 3/3
OF TECH CAMBRIDGE RESEARCH LAB OF ELECTRONICS W P DOVE
JUN 86 TR-518 N00014-81-K-0742

UNCLASSIFIED F/G 17/2 NL




1.0
1.1
1.25
1.4
1.6
1.8
2.0
2.2
2.8
3.2
3.6
4.0

CHAPTER 6

KBSP

6.1. Introduction

The previous chapters have presented the PDA system from several viewpoints. First from the perspective of the knowledge it contained and the concepts of its subsystems, then with an emphasis on implementation, and finally with a demonstration of its performance. In this chapter we discuss what we have learned from this thesis about combining numerical and symbolic processing in systems and what we have learned from other aspects of knowledge-based signal processing. Each section discusses some aspect of KBSP that relates to this thesis with comments on how that issue influenced the thesis or vice-versa.

6.2. Quantitative versus Qualitative Symbols

The value of an attribute can be numerical (1.5) or symbolic ("chocolate"). Consider the phrase "symbolic value" to mean any non-numerical value that has no obvious mapping into a number. For example, "chocolate" is a symbolic value but "five" is not. Within the subset of values that are symbolic, some quantify their attribute and others do not. For example, "chocolate" doesn't quantify the attribute "taste", since there is no obvious ordering of tastes. However, "high" does quantify any attribute it applies to[1], since there is an ordering between for example "high" and "low".

[1] Except when applied to a state of mind.

For "quantitative" symbolic values, there is always an ordering and there may be a metric as well. For example, in cooking one can substitute two "medium" peppers for one "large" one. However, it is rarely the case that arithmetic can be used on quantitative symbolic values. Usually, one recognizes the significance of a particular symbolic value. In the PDA, if the signal-to-noise ratio is "high", then silent regions can be detected by looking for places with "low" power.

Numerical values are not meaningful unless their scale can be determined or if they are simple counts ("the number of poles is 5"). The scale might be evident from context - "The first car took 6.7 seconds and the second car took 5.2." or standard usage - "He was doing 75 when the cop pulled him over". Some symbolic values do not need such a scale to be meaningful. Values like "voiced" can be interpreted solely based on their identity. Others derive their meaning from their identity and the attribute they are applied to (e.g. "stop" can be the value of "*phoneme identity*" or "traffic signal state"). Symbolic values that quantify their attribute do need a scale for them to be meaningful. For example, the value "high" needs some sort of scale. Sometimes a standard scale is implied by the attribute (e.g. "the stove is on high"). If not, a scale must be defined.

In a system that uses symbols to quantify numerical attributes, the scale is chosen at the time of quantification. For example, in the PDA the attribute SNR is "high" if it is larger than 40 dB and "low" otherwise. In performing this mapping from numbers to symbols, one divides the numerical range into symbolically labelled, mutually exclusive, exhaustive sets, reducing a near infinite number of possible numerical values to a small number of symbolic ones.

In many KBSP systems[67], all numerical attributes are quantified to a coarse symbolic range (e.g. "very high", "high", "low", "very low") before being used for any other purpose. Based on our experience with the PDA, this is not the best approach to building KBSP systems.

Mapping from a detailed description to a simplified one irreversibly loses information. This is only appropriate if the lost information is irrelevent or at least unnecessary. This is true for the Gold-Rabiner pitch detector, since the periodicity in speech is apparent in the extrema of a low-pass filtered version. It can also be true when one maps numerical values to symbolic quantitative ones for any particular purpose. An example of this is the SNR mapping in the PDA. This is only used to control whether or not to make silence_assertions based on power measurements.

However, such mappings are usually sensitive to the purpose to which the information is to be put. The extrema of a speech signal would be insufficient for purposes of reconstruction. Similarly, the numerical to symbolic mapping appropriate for one purpose may not be appropriate for another. If a system is sufficiently simple that a given mapping is only used for compatible purposes, then mapping the numbers "up front" will probably suffice. However, as that system grows, such mappings are likely to impede the inclusion of new knowledge since they will not be appropriate in general.

Some possible reasons for this use of initial numerical/symbolic mapping are:

1 The authors view the conversion from a high resolution numerical value to a low resolution (but still quantitative) symbolic value as retaining sufficient information about the attribute for their purposes.

2 These systems are built on top of rule systems that encourage or require the use of values taken from a small set. Hence the use of quantitative symbols taken from a small set rather than numerical values taken from a much larger one.

A rule system that relies on matching promotes the initial transformation from numbers to symbols. When using such systems, numerical values are less attractive because matching is impractical (with numerical values the number of possibilities is too large). With numerical values, other condition evaluation methods are required (e.g. the functional tests of the PDA).

An alternative to an initial transformation from numbers to symbols is for each rule to perform the mapping in its conditions. In that case, the mapping can be made in the specific context of the analysis to be performed (the conditions of that particular rule), and it places no restriction on the use of the (numerical) value of the attribute by other rules. Thus, many different applications for the same information can be easily supported, and the initial implementation of the system is not constraining future growth.

Based on this discussion, we have two pieces of advice about building KBSP systems:

1 Avoid an initial, single numerical-to-symbolic mapping unless it obviously preserves all the important information.

2 If the system is based on a rule system, make sure the rule system supports the convenient and efficient application of functional tests in rule conditions (e.g. the PDA rule system).

6.3. Symbolic/Numerical Interaction

One of the primary goals of this thesis was a better understanding of how to combine numerical and symbolic processing. In our work on the PDA, we employed four types of interaction between symbolic and numerical information:

Combination:
Numerical and symbolic information contribute to the same conclusion.

Verification:
Numerical information verifies symbolic assertions.

Supplying Parameters:
Symbolic information supplies advice for numerical processing.

Invocation Control:
Symbolic information determines where numerical processing will be applied.

Combination

In the PDA, two examples of the use of "combination" are the determination of the confidence in symbolic assertion VOICED and the determination of the statistics of the numerical assertion FINAL-PITCH. In the first case, there are symbolically derived assertions (phonetic-voiced, phonetic-frication, numerical-silence, burst) which add or deduct support for VOICED and thereby influence the voicing decision in the pitch track. In the second case, phonetically derived f0 estimates and numerically derived f0 estimates are combined to form the f0 probability densities of FINAL-PITCH which in turn determines the final pitch estimate of the PDA.

It is interesting to note that both interactions are accomplished numerically (through the merging of confidence estimates or the merging of probability density estimates). The (symbolic) identity of a supporter of VOICED serves to determine the polarity of the confidence contribution (phonetic-frication refutes VOICED whereas phonetic-voiced supports it). With f0 assertions, the (symbolic) identity of the phoneme and the sex/age (male, female, child) select the parameter values of the probability density estimates for f0 (mean, variance and odds) which is contributed to FINAL-PITCH.

Another example of the direct combination of symbolic and numerical information is in the actions of the Epoch system. Phoneme boundaries and boundaries of

numerically determined voiced segments both give rise to epochs which are then merged. Note also that the merging process itself is both numerical (the epochs must be close) and symbolic (the starting and ending epoch of a single assertion cannot be merged even if it would be numerically reasonable to do so).

In the PDA, the nature of the "combination" of objects (support versus denial, merging versus not) involves both numerical and symbolic information. The results of the combination can also be symbolic (newly established linkage between voicing marks caused by an epoch merge) or numerical (a new value for the f0 probability density or voicing confidence).

Verification

This type of interaction occurs when there are ways to verify or check an assertion (relatively) independently. For example in the PDA, the phoneme-marks in the input transcript might be erroneous (the phonetician makes a mistake). Also, the translation from a phoneme-mark to a voicing assertion may be erroneous (phonemes do not always manifest their typical acoustic properties). To check phonetic voicing assertions the PDA scans the waveform for phenomena that should accompany them. For example, a stressed phonetic-voiced mark is expected to have substantial low-frequency power near the center. If the measurements check out, the voicing-marks are given added support, otherwise support is deducted.

Another example of verification is in the use of numerical information in the PREMISE-ODDS form of symbolically derived assertions. For example, in the assertion of phonetic-aspiration and phonetic-frication from the phonetic marking of a stop, the PREMISE-ODDS (which provides support for the assertions) includes a comparison of the duration of the stop with the expected duration of stops of that type. If the

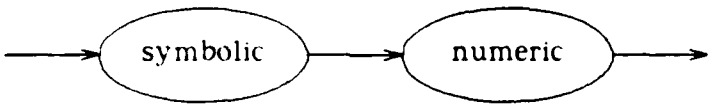
stop is marked with a length that substantially deviates from the typical duration, then the phonetic voicing assertions derived from it will only be weakly supported by the rule.

Supplying Parameters

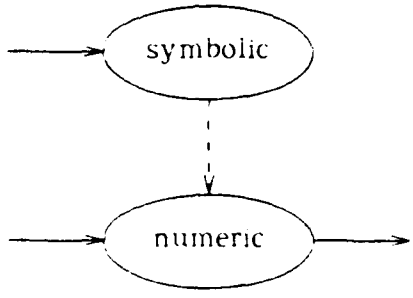
Information that results from one analysis can be used to provide "advice" for some other analysis to improve its performance. An example of this sort of interaction is in the use of the phonetic context to guide numerical f0 determination. For regions where the phonetic context can suggest an expected direction of similarity or an expected moveout, the PDA runs the numerical pitch detector with advice about what to expect. Given this advice, the pitch detector downgrades the confidence in results that do not conform to these expectations.

The PDA lacks any self-contained symbolic procedures like the numerical pitch detector. However, one can conceive of KBSP problems in which such interaction is plausible. If there was a symbolic procedure for parsing a stream of phoneme estimates into possible words, and a numerical procedure had determined that there was substantial high frequency noise, then the symbolic procedure could base its analysis primarily on the more robust voiced phonemes, be "suspicious" of the fricative phonemes, and look for errors involving voiced -> unvoiced phoneme substitutions.

Finally, one need not view supplying parameters as a rigid structure in the system. Such a view might depict symbolic analysis supplying parameters to numerical analysis in the following way



with the parameters being a required input to the numerical processing. Our view of this process, and the reason we used the word "advice", is that this information is optional and might be best depicted in the following way



with the dotted line from the symbolic to the numerical module signifying that information may or may not be forthcoming along that path.

Invocation Control

In the PDA, symbolic context may suggest certain numerical processing. For example, consider the case of numerical burst analysis. When an unvoiced stop is identified in the transcript, a numerical procedure is invoked there to find the boundaries of the stop burst.

This numerical procedure is only used in parts of the utterance where a burst is expected. Thus one need not be as careful in the design of the algorithm as would be the case if it were to be run everywhere. The symbolic invocation serves to "pre-filter" the speech eliminating regions which might cause erroneous results.

In a problem involving more complex symbolic analysis, it is easy to see where invocation of symbolic processing might be controlled by numerical information. In examining an image to identify objects (say automobiles) it might be the case that some simple numerical analysis (median filtering, local bandwidth, ...) might suggest locations where an automobile is likely. The symbolic analysis for the structure of

the automobile can then take place in those locations. This analysis might lead to erroneous results if applied indiscriminately.

Besides the potential advantage of avoiding false detection by only applying a procedure where appropriate, KBSP systems can be made more efficient by avoiding the application of complicated/expensive procedures of one type (symbolic or numerical) by using simple processing of the other type.

One final comment about invocation control. It is like and unlike the notion of "islands of certainty" as expressed in the HEARSAY system. As applied by HEARSAY to the problem of parsing speech, islands of certainty suggests that one start parsing in places where single words or short phrases are known with relative certainty and work outward from there. It suggests that the correct final parse will be likely to contain these confident words, and that one can get to the correct final parse more quickly by building outwards from such islands rather than constructing the entire parse in a piecemeal fashion.

Like islands of certainty, invocation control specifies where good places to work might be located. However, unlike islands of certainty, those places are not determined by the certainty in the information already present there. They are determined by properties present in those areas because of other analysis, either symbolic properties (as in the PDA when the transcript contains a stop) or numerical ones (as in the hypothetical example above).

The concept of islands of certainty suggests that places nearby might be preferable to work on. Our concept of invocation control makes no such statement. Thus, while both islands of certainty, and our concept of invocation control both involve controlling the places where analysis should take place, the former is an important

idea for problems with a certain structure (e.g. parsing speech) and the latter is important as a means for combining different types of information in any problem.

6.4. Combining Results

In a system that uses a parallel collection of modules to accomplish some task, some means must be provided for combining their results. The PDA uses this technique of multiple contributors for problems that involve both symbolic and numerical answers.

Numerical Information

A particularly good example of combining numerical results is the parallel processing Gold-Rabiner pitch detector. It uses six separate streams of extrema derived from the speech signal. Each is processed with a separate period detector to produce six sequences of period estimates, one estimate from each detector for each frame (100ms) of speech. Every 100 ms the "best" of the six most recent period estimates is chosen as the final result (if none are good enough, the frame is declared unvoiced).

Here are several points about the G-R pitch detector's information combination method:

Supporters are Equal

> In the G-R system, there are 36 potential supporters for each of the six period candidates. These supporters come from the period estimates of the current and two previous frames, and sums of pairs and sums of triples of these estimates. The rationale for using this set of supporters is unimportant for this discussion. The significant point is that each supporter is entitled to one vote in "choosing" the winner. Neither the origin of the supporter (e.g. whether it is a single period estimate, a sum, or a triple) nor any other information about the supporter (e.g. its reliability) is used to weight the votes.

Fixed Tolerance (prior assumption)

> The tolerance for supporters in G-R is fixed. This constitutes a prior assumption about the statistics of the supporters (if a candidate period is

correct, the assumption is that the supporters will be within a specified percentage of the candidate period). The G-R algorithm does not react to the posterior statistics of the supporters.

Hard Decision Boundaries

The confidence added by the vote of a single supporter is an all or nothing thing. It is either within the tolerance or not, any other information about proximity is discarded.

The PDA method for combining numerical information was not intended to be a replacement for the one used in G-R. It was intended as a model for how such information could be combined in any system. However, since G-R is an example of how that has been accomplished in the past, it is illuminating to compare the two.

The PDA method of combining numerical information employs the explicit probability densities associated with numerical assertions and uses the algebra of probability to combine them.

Support is Weighted

Each contributor to FINAL-PITCH supplies a gaussian density estimate with odds (effectively false alarm probability). Thus the contributions are weighted by their confidence, which is in turn determined from the quality of their source information.

No Fixed Tolerance or Prior Assumption

Each contributor establishes the treatment of their contribution by the statistics (mean and variance) they declare. There is no prior assumption in the combining method about the statistics of the supporters. In each run of the system, the statistics are redetermined.

No Hard Decision

The contribution of support is not an all or nothing thing. The closer a supporting assertion is to a candidate pitch, the more support it gives.

The ability to employ explicit statistical information makes this new approach to combining numerical information well suited to systems that must accept a wide variety of sources of numerical information, since each source can specify statistics that are appropriate for its own results (as is the case of the phonetically derived and

numerically derived f0 assertions in the PDA). Because it specifies the statistics of its results (i.e. the probability density for the numerical value in question), this method should be very useful in situations where the results must be further combined by the system. Also, a system that supplies such statistics in its output can be more helpful to a user than one that does not.

Symbolic Information

The approach used for combining symbolic values in the PDA corresponds closely with the one used by the PROSPECTOR system[56]. Both the PDA and PROSPECTOR express a probabilistic confidence in their assertions. Both use a combination rule derived from Bayesian probability theory, and both propagate confidence through a network connecting the assertions, so changes in confidence are immediately reflected throughout the network.

The distinctions between them are as follows:

Dimensionality of an Assertion

PROSPECTOR assertions corresponded to scalars. An assertion like "rock type=igneous" had a scalar confidence associated with it. In the PDA, many assertions are distributed in time and therefore correspond to vectors. An assertion such as "voicing=voiced" has a vector confidence which specifies the confidence independently for each sample covered by the assertion.

This distinction allows the PDA to have a modest number of assertions (in the hundreds) while responding to the information available with precision down to the sample level. This preserves fast rule system operation without sacrificing responsiveness to small events. The programming convenience and processing power of the KBSP package was crucial to this part of the implementation.

Absolute versus Relative Odds

PROSPECTOR assigns each of its assertions an absolute odds (equivalent to a probability of truth) and defines support from one assertion to another with two parameters (as discussed in chapter 4). The PDA assigns each assertion an odds-factor (the ratio of the current odds to the prior odds) and defines the support from one assertion to another in terms of probability of detection and probability of false alarm.

As was discussed in chapter 4, these changes eliminate the piecewise linear interpolation of probability used in PROSPECTOR, and thereby improve computational efficiency. Two other possible advantages are that the elimination of interpolation improves performance, and that rule writers familiar with concepts such as probability of detection might find it easier to express themselves using the PDA approach. However, these points are both conjectural. While odds-factors worked well on this problem, it is too early to know if this more "theoretically correct" procedure is in fact a practical improvement over PROSPECTOR's approach.

6.5. Alternate Values

There are occasions when contradictory data arise during processing. One way to deal with this situation is to immediately discard all but one of the conflicting results. Another is to keep around the alternatives and delay the choice (assuming it will subsequently become clear which result to chose).

For symbolic values such as "voiced" and "unvoiced", the contradictory data are competing assertions. The PDA allows contradictory assertions to coexist. All the support from such assertions is tallied via the unique VOICED assertion. The final voicing decision is made from its odds-factor. This is a fairly conventional approach which is often employed when assertions have explicitly recorded confidence.

For some numerical problems, there can be multiple values which do not represent conflict. This is the case with epochs in the PDA. Epochs are merged when they are symbolically compatible (they are not the start and end of the same assertion) and they are numerically compatible (their statistics admit to the possibility that they both pertain to the same underlying event). When epochs do not merge, they are simply kept separate, in effect implying that an additional underlying event has been detected.

For other numerical problems, conflict is an issue. In the case of f0 estimates a single number is desired. In the PDA we see a new possibility for dealing with conflicting values. The f0 probability densities of the assertion FINAL-PITCH combine the contributions of all f0 assertions into a single representation. The competing assertions are neither kept separate, nor are any discarded. In this way the multiple alternatives are all coexisting, but not as individuals. In effect the combined probability density represents a myriad of assertions about the potential value of f0, where the density at each choice of f0 represents the likelihood of that point being correct. The algebra of probability allows the system to manipulate this set of assertions as a group.

With this approach, it is necessary to use a sampled version of the f0 probability density because the combined density no longer conforms to any simple mathematical model. A single, sampled probability density for a numerical quantity takes the place of a set of alternative values for a symbolic quantity each with a specified confidence.

6.6. Mapping Numerical Values to Confidence

An important aspect of numerical/symbolic interaction in the PDA is the conversion of numerical value information into symbolic confidence information. Three cases of this are:

- The confidence of phonetic-frication assertions and phonetic-aspiration assertions (produced by rules that respond to stop environments) depends on the match between the duration of the marked stop and the typical duration of stops of that type. The more the marked stop duration differs from expectation, the lower the confidence in the phonetic voicing assertions.
- The confidence in numerical voicing assertions depends on the measured power within their domain. For example, a numerical-voiced assertion is created when low-frequency power is substantially above the background noise level. The greater that power, the more confident the numerical-

voiced assertion.

- The confidence in VOICED depends (in part) on the similarity (not the periodicity) of the waveform. If the waveform is substantially similar at some lag (from 2 to 20ms) then the confidence of VOICED is enhanced.

The above are examples where numerical values influence symbolic confidence. There is also one example where symbolic confidence converts to numerical value. That case is in the f0 decision made for generating a pitch track. A variation in the confidence of a supporting phonetic f0 assertion could cause a change in the position of the highest peak in the f0 probability density of FINAL-PITCH and thereby affect the final pitch choice.

All these examples are predicated on symbolic assertions having a numerical side (namely their confidence). The numerical value of some assertions is influencing the value of the symbolic confidence which is itself numerical. Despite the fact that it is implemented numerically, this is a case where a numerically valued assertion is having an impact on a symbolic one, and it is likely to be a common phenomenon in any KBSP system in which symbolic values are uncertain.

6.7. *Use of Explicit Statistics for Assertions*

The association of a confidence with each symbolic assertion has become a commonplace feature of symbolic processing systems. It provides a way for the system to process uncertainty as it processes values. This allows the system to model uncertain analyses accurately and to express this uncertainty to the operator. It can also serve system modularity by keeping together a value and the uncertainty about the value, rather than having the uncertainty explicitly or implicitly built into code that uses the value.

In the PDA, this idea is extended in two ways. First, the concept of explicitly associating a confidence with a data value is extended to numerical values; and second, the association of confidence with symbolic values is applied to distributed symbolic values in the form of a confidence sequence.

It is not uncommon in signal processing for the published results of an experiment to show confidence intervals on the values. Also, the notion of the statistics of an estimate is pervasive in the mathematics of signal processing. However, in our experience with signal processing systems, such statistical information is not carried along and processed with signal values, nor is it present in the outputs of signal processing systems (except when the primary purpose of the system is to compute the statistics and not the estimate).

In the PDA, the extension to numerical values is embodied in the statistics used for epochs and f0 estimates. While the details of these two applications differ, fundamentally they both demonstrate that explicit statistics associated with numerical values is a practical alternative to having the statistics of numerical values implicitly or explicitly encoded in the procedures that operate on those values (as in the G-R voting matrix).

The expression of time varying confidence for distributed symbolic assertions with sequences is an idea that is very helpful when the problem extends substantially along some dimension (e.g. time, frequency, space) and there are abstract entities that cover substantial intervals (e.g. phonemes, picture patches). By using a confidence sequence, one can avoid the necessity of attributing equal likelihood to an assertion over its entire domain.

6.8. Rule Conditions: Match versus Function

In contemporary rule based inference systems, there are two different approaches to determining the applicability of rules: matching and functional testing. Both are means to determine if a given set of circumstances warrant the application of a particular rule. In addition, the use of variables in matching provides a means of binding variables for use in other parts of the condition and in the actions of the rule.

Matching is a process that involves comparing a composite object representing a rule condition (a "pattern") with a similar object representing an assertion (a "text"). The pattern consists of literals and wildcards with variables to be bound to the wildcard components. The text is purely literals (while some systems use wildcard assertions for quantification, that issue is not relevant to this discussion). A typical rule will have several such patterns.

The rule is activated when there is a set of assertions such that the text literals match the pattern literals term for term, and all pattern wildcards with the same variable match identical text literals. For example, if wildcards in patterns and their associated variables were written with a leading hyphen, then figure 6.1 would be a

```
patterns
(phoneme -x end -e1)
(phoneme -y start -e2)
(simple-epoch -e1 composite -e3)
(simple-epoch -e2 composite -e3)

texts
(phoneme phoneme-mark-23 start simple-epoch-11 end simple-epoch-17)
(phoneme phoneme-mark-29 start simple-epoch-21 end simple-epoch-27)
(simple-epoch simple-epoch-17 composite composite-epoch-6)
(simple-epoch simple-epoch-21 composite composite-epoch-6)
```

Figure 6.1 Conditions done with Matching

condition that looked for consecutive phoneme marks (and a set of matching assertions)[2].

The distinction between functional tests and matching is that a specific procedure is invoked to test for satisfaction of a part of the condition, rather than a general procedure to compare a pattern with the text of the assertion. There are two examples of functional tests that can be compared to figure 6.1. The first (and most straightforward) is shown in figure 6.2a.

One significant difference in using functions is that one no longer gets binding as an automatic side effect (by matching wildcards). The "type" clauses provide the replacement for wildcard binding. They cause all assertions of the given type (i.e.

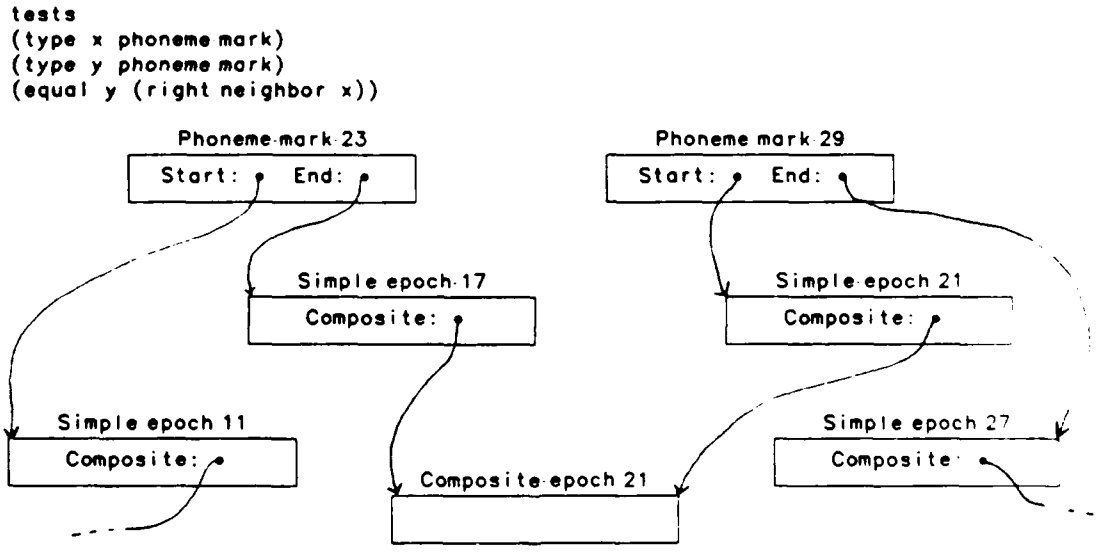


Figure 6.2a Conditions done with Functions #1

[2] As was discussed in chapter 4, the actual implementation of linkage between assertions in the PDA is with the simple-epochs that actually bound the assertion being connected to a common composite-epoch. Thus the need for the final two patterns to "extract" the composite for comparison.

phoneme-mark) to be tried as a choice for the specified variable (e.g. x). The sole test is (*equal y (right −neighbor x)*).

Here we see one of the significant advantages of using functional tests. There is no longer any mention of the internal representation of phoneme-marks (i.e. epochs). Instead, we are able to employ the abstract function (*right −neighbor ...*) to extract the necessary information. In this way one can separate the expression in the rule condition from the structure of the assertion.

There is another somewhat subtler distinction between functional tests and matching that can best be shown by another version of the rule (see figure 6.2b). In this example there are no tests whatever. Instead, the value for the variable y is directly extracted from the network structure in which x is embedded. When functions are used in rule conditions, information can be extracted at any depth from

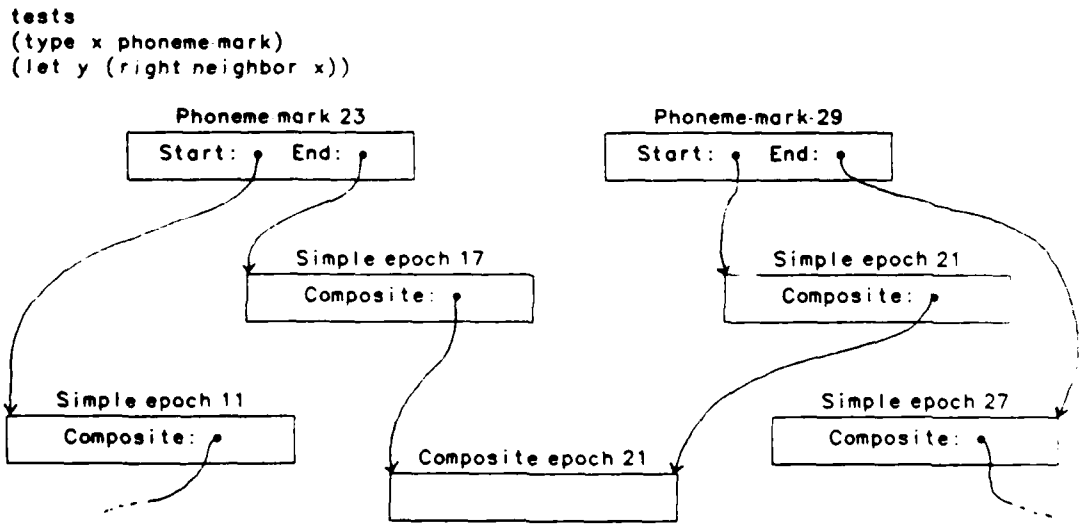


Figure 6.2b Conditions done with Functions #2

underlying data structures. With matching, a single pattern can extract information that is directly present in a single assertion (as the simple epoch values -e1 and -e2 are extracted by the first two patterns in figure 6.1). However, it requires additional patterns if the object of interest (e.g. the composite-epoch) is only indirectly related. These additional patterns come at a price (the cost of comparing them against all the assertions that they do not match). By using a function to extract the desired information one can avoid this wasted computation.

The advantage or disadvantage of match may depend on whether the data structure of the assertion texts is usable as the means of storing information in the system. When assertion texts are themselves the means of storing information (e.g. as in OPS-5), when there is no underlying (hidden) linkage to be exploited, and when isolating the structure of assertion texts from rules is deemed unimportant, then match can be an efficient and easily implemented approach to rule condition evaluation. In the PDA, the existence of complex hidden data relationships, the cost of creating assertion texts, the extra effort of implementing code for matching, and the desire to isolate the rule conditions from data structure all led to the decision to avoid using match in rule conditions.

Another case for the use of match is that other parts of the system can easily analyze the rule patterns for purposes such as the focusing of attention on particular rules during certain types of processing. It can be extremely difficult to automatically examine the (potentially arbitrary) expressions that may result from an unconstrained functional approach to rule conditions. We did not face this issue in the PDA since we did not employ such automatic analysis of the rules for control purposes. However, it seems that the crux of this issue may be in constraining the syntax of the rule and not

in the choice of a match or functional tests as a way of evaluating conditions.

Using functional tests for condition evaluation can be more costly than match. The dramatic efficiencies of the discrimination net (as used by systems such as YAPS and OPS-5) have not been achieved for functional tests. While some efficiency is possible (see chapter 4) and the use of underlying data structure may have computational advantages, there is still substantial cost in the use of functions for rule conditions, and this can be compounded by the added programming complexity.

The following point may be the most important with respect to KBSP. Match is an appropriate mechanism for determining equality. If the values of objects require complicated numerical tests to determine the applicability of a rule, then functions become a necessity. While it may be possible to use match with quantitative symbols for some numerical applications (as discussed previously), a programming environment for numerical applications should support the testing of numerical values in rules.

6.9. The Use of Dependency

The PDA demonstrates two ways in which dependency can be used in a KBSP problem: for propagating changes in statistics and for keeping the system consistent.

The PDA method of statistical propagation is based on the one used by the PROSPECTOR system. In that system, the assertions were all interconnected through a network that propagated a change in the odds of any assertion to all supported assertions. In PROSPECTOR, the assertions not distributed in time or space, and the statistics were just the confidence of the assertion. In the PDA this same idea is used for atomic symbolic valued assertions (as in PROSPECTOR), for distributed symbolic valued assertions (e.g. voicing), and for numerically valued assertions (e.g. f0). This ability

to keep statistical information up to date in the face of change is very helpful in an interactive system (e.g. an assistant) where the operator may volunteer information. It is also useful in cases where information is not supplied in a known order.

Contemporary rule systems do not retract information that was derived from rules that are no longer valid, or retract information based on object configurations that no longer exist. By using the ideas behind TMS[55], we implemented a history independent rule system. With such a rule system, rules and data can be entered and retracted in any order. The final results depend only on the rules and assertions which remain. While this may offer some useful potential for self modification on the part of the system, that mode of behavior was not used in the PDA. For the PDA, the biggest advantages of history independence were in the convenience of ignoring the ordering of rule and data addition, and in the ability to make (fast) incremental modification for test purposes without the need to reinitialize the system.

6.10. Comments

This chapter has made several suggestions about systems that combine symbolic and numerical knowledge.

- Mapping numeric values to symbolic values before processing is inadvisable if the mapping might not be appropriate for all applications.
- Symbolic and numerical information can be combined directly as equal contributors, one can verify the other, one can be used as advice for processing of the other type, and one can guide the places where processing of the other type is applied. Examples of all these types of interaction are present in the PDA.
- Information from symbolic and numeric sources can be combined either as numeric values (f0) or symbolic ones (voicing). We have presented effective new computational models for both of these situations.
- Information about the boundaries of symbolic assertions can be effectively processed with a system that uses both symbolic and numerical criteria for linking the information. Since such assertions can be derived from both numerical and symbolic sources, this too represents a means of combining

such information. We have demonstrated the effectiveness of such a system.

- The explicit representation of uncertainty makes a system more modular, allows the contribution of each assertion to be based on its quality, and allows the system to express its confidence to the user. This theme is evident in the Epoch system, and the Knowledge Manager, and the outputs of the PDA. We have employed new uncertainty representations for symbolic assertions (the odds-factor), distributed symbolic assertions (sequential confidence), and numerical assertions (the f0 density).
- When using numerically valued assertions, a rule system with functional tests is likely to be necessary. Such a rule system can also contribute to system modularity (through abstract rule conditions), and system efficiency (through the access of information in a network of assertions). We have implemented such a rule system with a novel means of efficiently evaluating functional conditions.
- In any rule based system, the ability to insert the rules and data in any order simplifies the behavior of the system and the task of operating it; the ability to add and remove rules and data facilitates development. Using the ideas behind truth maintenance systems, together with the idea of change propagation, we have developed a history independent rule system that allows both of these types of user interaction.

Our original goal was to learn more about how to build systems for problems that involved both symbolic and numerical knowledge. We have learned a great deal about ways to go about this task. The previous points are general statements about the task, and the earlier chapters have presented specific examples of how one such system was built. The PDA demonstrates that extensive interaction between symbolic and numerical knowledge is practical, and that it appears to be effective. In the last chapter we will discuss some ideas for future work on this area.

CHAPTER 7

Conclusions

7.1. Thesis Contributions

This thesis makes several significant contributions to Knowledge-Based Signal Processing. The specific points were outlined in the preceding chapter, but in general they concern methods of combining symbolic and numerical problem knowledge in systems. In addition to these contributions, there are specific contributions to signal processing, pitch detection and expert systems.

Signal Processing

This thesis defines the signal processing operation we call the Normalized Local Autocorrelation (NLA), an application of the idea behind the correlation coefficient to a single time sequence for the purpose of locating similarities. We investigated the effects of the window in this function and determined that the window should be chosen as the square root of a suitable band-limiting window (such as Hamming window). We also gave an efficient FFT based structure for computing the NLA, and finally, we pointed out that the NLA has the feature of being insensitive to exponential taper (a property that is very important in determining the period of signals with a time-varying envelope such as speech).

The signal representation developed for this thesis is the first to use an unbounded domain, together with random access of the signal values. Since one of the primary goals of such a representation is to capture the broadest possible class of

signals, the ability to represent signals with infinite domains that are both periodic and aperiodic is a significant advance. Another goal is the ability to express signal processing knowledge in the most straightforward fashion. Since this signal representation allows the convenient expression of concepts such as zero-phase and linear-phase, and since it can preserve the offset of signal sections that are involved in such operations as overlap-add convolution, it enhances our ability to program signals.

Pitch Detection

Our contribution to pitch detection takes two forms. First, we compiled a list of the knowledge that bears on the pitch detection problem. While Hess has compiled a much more exhaustive treatment of the signal processing part of this knowledge[29], we were unable to find a convenient reference for the wealth of speech knowledge. Chapter 2 serves as a compact source of references about the knowledge in this problem.

The other contribution made to pitch detection is in the theme of the PDA. Pitch detectors have always had a strong numerical flavor, and have not made substantial use of the knowledge about pitch that stems from speech research. While the PDA is not appropriate for many conventional pitch detection problems (transcripts are not always available), it does demonstrate that the information present in speech (about stress, phonetic identity, etc.) can be useful for pitch detection. This suggests that systems which are capable of deriving such information from the waveform would be able to make better use of the knowledge unique to speech.

Expert Systems

One of the points mentioned in the preceding chapter has implications that go beyond systems for symbolic/numerical problems. The feature of history independence in the rule system was profoundly useful in the development of the PDA system and the insertion of the speech knowledge that it used. That feature would be of value to any expert system project, whether numerical information was involved or not.

Another idea used in the PDA that is important for expert systems is the use of functions in rule conditions to extract information from a network of assertions. By not performing the implicit iteration that matching and unification require, this technique can save 90% or more of the work involved in determining neighbors and other information that is not explicitly present in any one assertion. This idea would apply to any problem which can be viewed as a large number of related entities, each of which must be interpreted based on its own properties and those of its neighbors.

7.2. Future Work

In the PDA there are networks for maintaining support and for maintaining logical dependency. In building the system we had to avoid making loops in these networks, or the propagation of information around them would never cease. It is not hard to envision a case where loops would make sense. For example, if a numerical parameter was estimated approximately and that approximation was used to refine the estimate of the parameter than a loop could result. Our method for propagating support and dependency information will not work with such loops, but there may be alternative propagation methods that would work.

In combining numerical assertions, the PDA started with assertions represented in parametric form (mean, deviation, and confidence) and combined them using a uniformly sampled sequence to represent the resulting density. Two areas for future work are:

- What is a good method for combining numerical assertions which are expressed in the form of arbitrary sequences rather than gaussian parameters? Such a method would be necessary if the results of the initial combination were to be used in a second stage of processing.
- What are alternative representations for such probability densities from which the true density can be determined? Uniformly sampled sequences can lose contributions that involve very narrow peaks.

Some of the knowledge in the PDA was compiled into the numerical pitch detector. While rules served as a convenient modular way of expressing knowledge, they were inefficient and awkward for programming the numerical pitch detector. Is there a way to make the rule system framework better for expressing such knowledge, or is that type of knowledge inherently difficult to express with rules?

Bibliography

[1] L. Erman, R. Hayes-Roth, V. R. Lesser, and D. R. Reddy, "The Hearsay-II Speech Understanding System: Integrating Knowledge to Resolve Uncertainty," *Computing Surveys*, vol. 12, pp. 213-254, Jun. 1980.

[2] H. P. Nii, E. A. Feigenbaum, J. J. Anton and A. J. Rockmore, "Signal-to-Symbol Transformation: HASP/SIAP Case Study," *AI Magazine*, vol. 3, pp. 23-35, Spring 1982.

[3] B. Gold and L. R. Rabiner, "Parallel Processing Techniques for Estimating Pitch Periods of Speech in the Time Domain," *JASA*, vol. 46, pp. 442-448, Aug. 1969.

[4] J. J. Ohala, "The physiology of tone," in *Consonant Types and Tone*, USC Occasional Papers in Linguistics, July 1973, pp. 3-14.

[5] J. L. Flanagan, C. H. Coker, L. R. Rabiner, R. W. Schafer, and N. Umeda, "Synthetic voices for computers," *IEEE Spectrum*, vol. 7, no. 10, pp. 22-45, Oct. 1970.

[6] L. R. Rabiner and R. W. Schafer, *Digital Processing of Speech Signals*. Englewood Cliffs, NJ: Prentice-Hall, 1978.

[7] A. E. Rosenberg, "Effect of glottal pulse shape on the quality of natural vowels," *JASA*, vol. 49, no. 2, pp. 583-590, Feb. 1971.

[8] J. Makhoul, R. Viswanathan, R. Schwartz, and A. W. F. Huggins, "A Mixed-Source Model for Speech Compression and Synthesis," *JASA*, vol. 64, no. 6, pp. 1577-1581, Dec. 1978.

[9] G. E. Peterson and H. L. Barney, "Control Methods Used in a Study of Vowels," *JASA*, vol. 24, no. 2, pp. 175-184, Mar. 1952.

[10] D. E. Veeneman and S. L. BeMent, "Automatic glottal inverse filtering from speech and electroglottographic signals ," *IEEE Trans. ASSP*, vol. 33, no. 2, pp. 369-376, Apr. 1985.

[11] D.J. Wong, J.D. Markel, and A.H. Gray, "Least squares glottal inverse filtering from the acoustic speech waveform," *IEEE-ASSP*, vol. 27, pp. 350-355, Aug. 1979.

[12] M. R. Matausek and V. S. Batalov, "A new approach to the determination of the glottal waveform," *IEEE Trans. ASSP*, vol. 28, no. 6, pp. 616-622, Dec. 1980.

[13] W. A. Lea, "Segmental and Suprasegmental Influences on Fundamental Frequency Contours," in *Consonant Types and Tone*, USC Occasional Papers in Linguistics, July 1973, pp. 17-69.

[14] I. Lehiste and G. E. Peterson, "Some Basic Consideration in the Analysis of Intonation," *JASA*, vol. 33, no. 4, pp. 419-425, 1961.

[15] A. S. House and G. Fairbanks, "The influence of consonant environment upon the secondary acoustical characteristics of vowel," *JASA*, vol. 25, no. 1, pp. 105-113, Jan. 1953.

[16] H. Fujisaki, "Dynamic Characteristics of Voice Fundamental Frequency in Speech and Singing," *Fourth F.A.S.E. Symposium*, 1981.

[17] H. Fujisaki, "Modeling the dynamic characteristics of voice fundamental frequency with applications to analysis and synthesis of intonation," *XIII International Congress of Linguists*, pp. 57-69, 1982.

[18] V. W. Zue, "Acoustic Characteristics of Stop Consonants: A Controlled Study," *Tech. Rpt.*, vol. 523, Lincoln Lab, 1976.

[19] D. H. Klatt, "Linguistic uses of segmental duration in English: Acoustic and perceptual evidence," *JASA*, vol. 59, no. 5, pp. 1208-1221, May 1976.

[20] D. H. Klatt, "Vowel lengthening is syntactically determined in a connected discourse," *Journal of Phonetics*, vol. 3, pp. 129-140, 1975.

[21] D. O'Shaughnessy, "Modelling Fundamental Frequency, and its Relationship to Syntax Semantics and Phonetics," PhD, MIT, 1976.

[22] W. E. Cooper and J. M. Sorenson, "Fundamental Frequency Contours at Syntactic Boundaries," *JASA*, vol. 62, no. 3, Sept. 1977.

[23] J. B. Pierrehumbert, "The Phonology and Phonetics of English Intonation," PhD, MIT, 1980.

[24] D. P. Huttenlocher and V. W. Zue, "A model of lexical access from partial phonetic information," *Proc. ICASSP*, Mar. 1984.

[25] M. Liberman and J. B. Pierrehumbert, "Intonational Invariance Under Changes of Pitch Range and Length," BLTJ, 1982.

[26] Y. Horii, "Fundamental frequency perturbation observed in sustained phonation," *Journal of Speech and Hearing Research*, vol. 22, pp. 5-19, Mar. 1979.

[27] Y. Horii, "Vocal shimmer in sustained phonation," *Journal of Speech and Hearing Research*, vol. 21, no. 1, pp. 202-209, 1980.

[28] L. Dolansky and P. Tjerlund, "On Certain Irregularities of Voiced-Speech Waveforms," *IEEE-AU*, vol. AU-16, no. 1, March 1968.

[29] W. Hess, *Pitch determination in speech signals*. Berlin: Springer-Verlag, 1983.

[30] J. L. Flanagan, *Speech Analysis Synthesis and Perception*. New York, NY: Springer-Verlag, 1972.

[31] J. J. Dubnowski, R. W. Schafer, and L. R. Rabiner, "Real-Time Digital Hardware Pitch Detector," *IEEE Trans. ASSP*, vol. 24, no. 1, pp. 2-8, Feb. 1976.

[32] M. M. Sondhi, "New Methods of Pitch Extraction," *IEEE Trans. AU*, vol. AU-16, no. 2, pp. 262-266, June 1968.

[33] J. D. Markel, "The SIFT algorithm for fundamental frequency estimation," *IEEE Trans. Audio and Electroacoustics*, vol. 20, no. 5, pp. 367-377, Dec. 1972.

[34] L. R. Rabiner, M. R. Sambur, and C. E. Schmidt, "Applications of a nonlinear smoothing algorithm to speech processing," *IEEE Trans. ASSP*, vol. 23, no. 6, pp. 552-557, Dec. 1975.

[35] M. J. Ross, H. L. Shaffer, A. Cohen, R. Freudberg, and H. J. Manley, "Average magnitude difference function pitch extractor," *IEEE Trans. ASSP*, vol. 22, no. 5, pp. 353-362, Oct. 1974.

[36] N. J. Miller, "Pitch detection by data reduction," *IEEE Trans. ASSP*, vol. 23, no. 1, pp. 72-79, Feb. 1975.

[37] S. Seneff, "Real-Time Harmonic Pitch Detector," *IEEE Trans. ASSP*, vol. 26, no. 4, pp. 358-365, Aug. 1978.

[38] J. A. Moorer, "The optimum comb method of pitch period analysis of continuous digitized speech," *IEEE Trans. ASSP*, vol. 22, no. 5, pp. 330-338, Oct. 1974.

[39] R. J. Sluyter, H. J. Kotmans, and T. Claasen, "Improvements of the harmonic-sieve pitch extraction scheme and an appropriate method for voice-unvoiced detection," *Proc. ICASSP*, pp. 188-191, 1982.

[40] H. Duifuis, L. F. Williams, and R. J. Sluyter, "Measurement of pitch in speech: an implementation of Goldstein's theory of pitch perception," *J. ASA*, vol. 71, no. 9, pp. 1568-1580, June 1982.

[41] T. V. Sreenivas and P. V. S. Rao, "Pitch extraction from corrupted harmonics of the power spectrum," *JASA*, vol. 65, no. 1, pp. 223-227, Jan. 1979.

[42] M. Piszczalski and B. A. Galler, "Predicting musical pitch from component frequency ratios," *JASA*, vol. 66, no. 3, pp. 710-720, Sept. 1979.

[43] M. R. Schroeder, "Period Histogram and Product Spectrum: New Methods for Fundamental-Frequency Measurement," *JASA*, vol. 43, no. 4, pp. 829-833, Jan. 1968.

[44] A. M. Noll, "Pitch determination of human speech by the harmonic product spectrum, the harmonic sum spectrum, and a maximum likelihood estimate," *Proc. Symposium on Computer Processing in Comm.*, pp. 779-797, April 8-10, 1969.

[45] A. M. Noll, "Cepstrum Pitch Determination," *JASA*, vol. 41, no. 2, pp. 293-309, Aug. 1966.

[46] D. Friedman, "Pseudo-Maximum-Likelihood Speech Pitch Extraction," *IEEE Trans. ASSP*, vol. 25, no. 3, pp. 213-221, June 1977.

[47] J. D. Wise, J. R. Caprio, and T. W. Parks, "Maximum Likelihood Pitch Estimation," *IEEE Trans. ASSP*, vol. 24, no. 5, pp. 418-423, Oct. 1976.

[48] J. N. Maksym, "Real-time pitch extraction by adaptive prediction of the speech waveform," *IEEE Trans. Audio and Electroacoustics*, vol. 21, no. 3, pp. 149-154, June 1973.

[49] J. D. Markel, "Application of a digital inverse filter for automatic formant and f0 analysis," *IEEE Trans. Audio and Electroacoustics*, vol. 21, no. 3, pp. 154-160, June 1973.

[50] D. T. L. Lee, "A novel innovations based time-domain pitch detector," *Proc. ICASSP*, pp. 40-44, 1980.

[51] B. S. Atal and L. R. Rabiner, "A pattern recognition approach to voiced-unvoiced-silence classification with applications to speech recognition," *IEEE Trans. ASSP*, vol. 24, no. 3, pp. 201-212, June 1976.

[52] V.V.S. Sarma, "Studies in pattern recognition approach to voiced-unvoiced-silence classification," *Proc. ICASSP*, pp. 1-4, 1978.

[53] J. F. Allen, "Maintaining knowledge about temporal intervals," *Comm. of the ACM*, vol. 26, no. 11, pp. 832-843, Nov. 1983.

[54] W. J. Long, "Reasoning about state from causation and time in a medical domain," *Proc. AAAI*, pp. 251-254, 1983.

[55] D. A. McAllester, "A three valued truth maintenance system," A.I. Memo 473, MIT AI Lab, Cambridge, MA., May 1978.

[56] R. O. Duda, P. E. Hart, N. J. Nillson, R. Reboh, J. Slocum, and G. L. Sutherland, "Development of a computer-based consultation for mineral exploration," *Annual Report SRI Projects 5821 and 6415*, SRI Intl., 1977.

[57] G. E. Kopec, "The representation of discrete-time signals and systems in programs," PhD, MIT, 1980.

[58] C. Myers, "Numeric and Symbolic Representation and Manipulation of Signals," PhD, MIT, Cambridge, MA., 1986.

[59] G. E. Kopec, "The signal representation language SRL," *Proc. ICASSP*, 1983.

[60] R. O. Duda and P. E. Hart, *Pattern Classification and Scene Analysis*. New York, NY: Wiley-Interscience, 1973.

[61] E. Allen, " "YAPS: Yet another production system"," *Proc. AAAI*, pp. 5-7, 1983.

[62] C. L. Forgy and J. McDermott, "OPS, A domain-independent production system language," *Proc. Joint Conference on Artificial Intelligence*, pp. 933-939, 1977.

[63] C. V. Kimball and T. L. Marzetta, "Semblance processing of borehole acoustic array data," *Geophysics*, vol. 49, no. 3, pp. 274-281, Mar. 1984.

[64] D. Griffin, "A New Speech Model and its Use in Speech Coding," PhD, MIT, Cambridge, MA., 1986.

[65] H. L. Van Trees, *Detection, Estimation, and Modulation Theory, Part III*. New York: John Wiley & Sons, 1971.

[66] C. A. McGonegal, L. R. Rabiner, and A. E. Rosenberg, "A Semiautomatic Pitch Detector (SAPD)," *IEEE Trans. ASSP*, vol. 23, pp. 570-574, Dec. 1975.

[67] A. M. Nazif and M. D. Levine, "Low level image segmentation: an expert system," *IEEE Trans. PAMI*, vol. 6, no. 5, pp. 555-577, Sep. 1984.

DISTRIBUTION LIST

	DODAAD Code	
Director Defense Advanced Research Project Agency 1400 Wilson Boulevard Arlington, Virginia 22209 Attn: Program Management	HX1241	(1)
Head Mathematical Sciences Division Office of Naval Research 800 North Quincy Street Arlington, Virginia 22217	N00014	(1)
Administrative Contracting Officer E19-628 Massachusetts Institute of Technology Cambridge, Massachusetts 02139	N66017	(1)
Director Naval Research Laboratory Attn: Code 2627 Washington, D.C. 20375	N00173	(6)
Defense Technical Information Center Bldg 5, Cameron Station Alexandria, Virginia 22314	S47031	(12)
Dr. Judith Daly DARPA / TTO 1400 Wilson Boulevard Arlington, Virginia 22209		(1)

END

DTIC

7-86